Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

Capstone Report:
Analysis of Calgary Police Service “X99 Miscellaneous” Calls

Summary

The Calgary Police Service (CPS) uses a Computer Assisted Dispatch (CAD) system for receiving incoming 911 calls and sharing information about the call with responding officers. CAD allows them to record information about the calls and classify calls according to their respective ‘event types,’ such as “robbery,” “domestic dispute,” or “noise complaint.” However, the CPS has observed a trend in recent years: there has been a significant increase in the categorization of calls under the event type "X99 Miscellaneous." This trend is problematic, as the classification of a substantial portion of calls as simply ‘miscellaneous’ is not informative for their analytics. The goal of our project is to explore the underlying reasons for the increase in miscellaneous calls and potentially identify common topics within these calls. This project would help CPS improve their call classification procedure.

To tackle this problem, we attempted three approaches: (1) **Topic modeling using Latent Dirichlet Allocation (LDA),** (2) **Sentence embedding & K-means clustering,** and (3) **Zero-shot topic modeling using BERTopic**. Through the methodologies employed, we derived clusters and identified topics that would be valuable for the CPS to investigate as potential sources of the increase in X99 Miscellaneous calls. To evaluate our findings, we checked the distribution of certain variables in each cluster. Additionally, we sought validation from the CPS, aligning our results with their domain expertise.

Our final round of clustering using BERTopic was performed with a minimum cluster size of 25 events, resulting in a total of 76 clusters. Of 9,752 total events in the dataset, 3,580 events (36.7%) could not be clustered alongside other calls and were therefore considered to be the most “truly miscellaneous” of the X99 Miscellaneous calls provided to us. We also generated word clouds for the ten most numerous clusters, which allowed us to present them to the CPS in an easily-interpretable format.

The following sections will detail the data we employed, the modeling methodologies we adopted and the insights we gained about model performance.

Data

Description of data:

During the first week of the project, the CPS needed to perform a more rigorous redaction of personal information than they had originally anticipated, so there was a slight delay in providing our group with the full data set they had intended to supply. Instead, they provided our group with a sample data file that would allow us to begin a

basic exploratory data analysis (EDA) and familiarize ourselves with the structure of the data. The sample data consisted of an Excel file with one week's worth of Computer Assisted Dispatch (CAD) calls, collected between January 1st and January 7th, 2023. This data consisted of 6,000 rows of call data, comprising 732 unique events. Each event could have multiple CAD calls associated with it.

During the second week of the project, the CPS finished their redaction process and supplied us with the full data set, which consisted of 72,034 CAD calls made between January 1st and March 30th, 2023. The calls detailed 9,752 unique events, all of which had received a final event type of X99 Miscellaneous.

Each row of the provided data included the following data fields:

Call_Created_Date	The date when the CAD event was first created.
Event_Anonymizer	Numeric identifier for each unique event, but modified by the CPS for the sake of privacy.
Grouped_Event_Type_Code_Desc	The final Event Type at the point when the event was closed. All provided rows were `Miscellaneous`.
Grouped_Event_Subtype_Code_Desc	The final Event Sub-Type at the point when the event was closed. All provided rows were 'X99 Miscellaneous.'
Occurrence_Type	Indicates if a corresponding occurrence report was entered into the Records Management System (RMS). If so, indicates the type of report. If not, None.
Occurrence_Type_UCR_Category	Type of report according to the Uniform Crime Reporting (UCR) program, which is administered by Statistics Canada. It provides a standard way to group crime reports.
Occurrence_Report_Category	Provides additional granularity to the UCR report category.
Priority	In Calgary, calls are dispatched according to Urgency (1=High, 2=Medium, 3=Low, 9=Officer already on-scene).
Public_Generated_Event_Flag	Whether or not the CPS believes the call was generated by a member of the public (TRUE) or by a police officer (FALSE).
Event_Attended_Flag	Whether the CAD system indicates a police unit arrived at-scene (TRUE) or not (FALSE).
Event_Remarks_Text	Text written by 911 operators or officers

	relating to the content of a call, including comments, description of history, nature of the incident, re-classification of the event type, etc. the CPS identifies this as the data field of interest.
Event_Remarks_Created_Timestamp	Date-timestamp when the text comment was added. Used for sorting the text comments.
Remarks_Line_Order	Number indicating intended reading order for lines which share the same event and same timestamp.
Remarks_Line_Grouping	Number distinguishing lines which share the same event and same timestamp from each other.

Each row corresponded to an event, identified by a unique Event_Anonymizer value. Events may be either generated by the public when someone phones 911, or generated by the police internally, either by an officer or 911 operator. Publicly-generated events are distinguished from internally-generated ones by the Public_Generated_Event_Flag, which can be ‘True’ or ‘False.’

Our partners at the CPS indicated that they had previously attempted some analysis of this data, but that they had not attempted any NLP-based analysis on the text data, located in the field Event_Remarks_Text. As such, while they indicated that our group was free to use any of the data fields in our analysis, the principal field of interest was to be the Event_Remarks_Text. Each text consisted of brief notes made by a police officer or 911 operator responding to a call. These notes typically outlined the nature of the event, specific details about the people involved and the scene, information reported by people interviewed, or requests for specific responses from the police, such as backup or a timer to check in on an officer’s wellbeing. Many calls included a police officer re-categorizing the event from a previously designated event type to the X99 Miscellaneous type, often when they had determined the case to be concluded or not a police issue.

The content of each text contains a lot of police jargon which can be challenging to interpret by people lacking domain knowledge such as us. This includes:

- Frequent use of abbreviations, e.g.
    - ‘veh’ for ‘vehicle’
    - ‘perp’ for ‘perpetrator’
    - ‘epo’ for ‘Emergency Protective Order’
- Police codes for identifying various crimes, and other dispatch codes, e.g.
    - ‘C138’

    - ‘D120’
- Technical phrases relating to police operations and logistics, e.g.
    - ‘Special Address Comment’
    - ‘Accept Advised Event’

Because of the high frequency of jargon used, it was necessary to refer to the capstone partners in order to interpret and evaluate the validity of our analysis, as domain knowledge of these terms is crucial for identifying whether or not calls are being grouped together in a logical way.

As our task was to investigate why the number of X99 Miscellaneous calls had increased since 2020, all of the call data provided to us consisted of calls which ended up with a final event type of X99 Miscellaneous. As such, we initially did not have any data displaying how non-miscellaneous calls were categorized. In order to get a better sense of this, we requested that our partners at the CPS provide us with data which demonstrated event types other than X99, which they did.

They provided us with additional data collected between January and March of 2023, which contained the following event types: ‘collision,’ ‘sexual offences,’ ‘check on welfare,’ ‘assault,’ ‘noise,’ 'medical,' 'harassment/threats,' 'theft,' 'alarm,' 'missing person,' 'break and enter,' 'abduction,' 'drugs,' 'property,' 'domestic,' 'traffic,' 'disturbance,' 'suspicious,' 'assistance,' 'mental health,' 'dispatch,' 'robbery,' ‘fraud,' and 'escaped.' These provided us with a good basis for building clusters, and recognizing what types of groups we might expect to form from the miscellaneous data.

Preprocessing:

As the raw data was presented to us with each event separated into multiple rows, we first pivoted the data, grouping it by Event_Anonymizer ID in order to ensure we had only one row per event. This way, we had one unified text field per event that we could pass as a single input.

Next, we removed stop words, including both common English stop words as well as a list of domain-specific stop words we procured from the CPS. This list included the terms *Advised, CAD, Caller, Chief, Comments, Complaint/CO/COMP, Description, Determinant, ECO, Event, Event accept, Incident, Problem, Says, Timer, X99,* and *Xref*. Our partners identified these terms as being likely to appear with equal frequency across all event types.

As many of the event texts contained police jargon in the form of abbreviations, we expanded many abbreviations into their full forms using a glossary. This was done to ensure consistency between all instances of a word.

Finally, To ensure we were capturing all information at our disposal, we also concatenated the fields Occurrence_Type, Occurrence_Report_Category, and Priority to the Event_Remarks_Text. This way, our analysis would capture a more holistic view of each row, rather than relying on the raw text alone.

Methods

In this section, we will detail the methods employed to explore and analyze the 911 call log data, leveraging various natural language processing (NLP) and machine learning techniques. We aimed to identify underlying themes and patterns in the call logs using three different approaches: (1) Topic modeling using Latent Dirichlet Allocation (LDA), (2) Sentence embedding & K-means clustering, and (3) Zero-shot topic modeling using BERTopic.

Workflow Overview:

The overall workflow for our analysis involved several key steps, from data preparation and preprocessing to model training and evaluation. The following diagram outlines the entire process:

Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

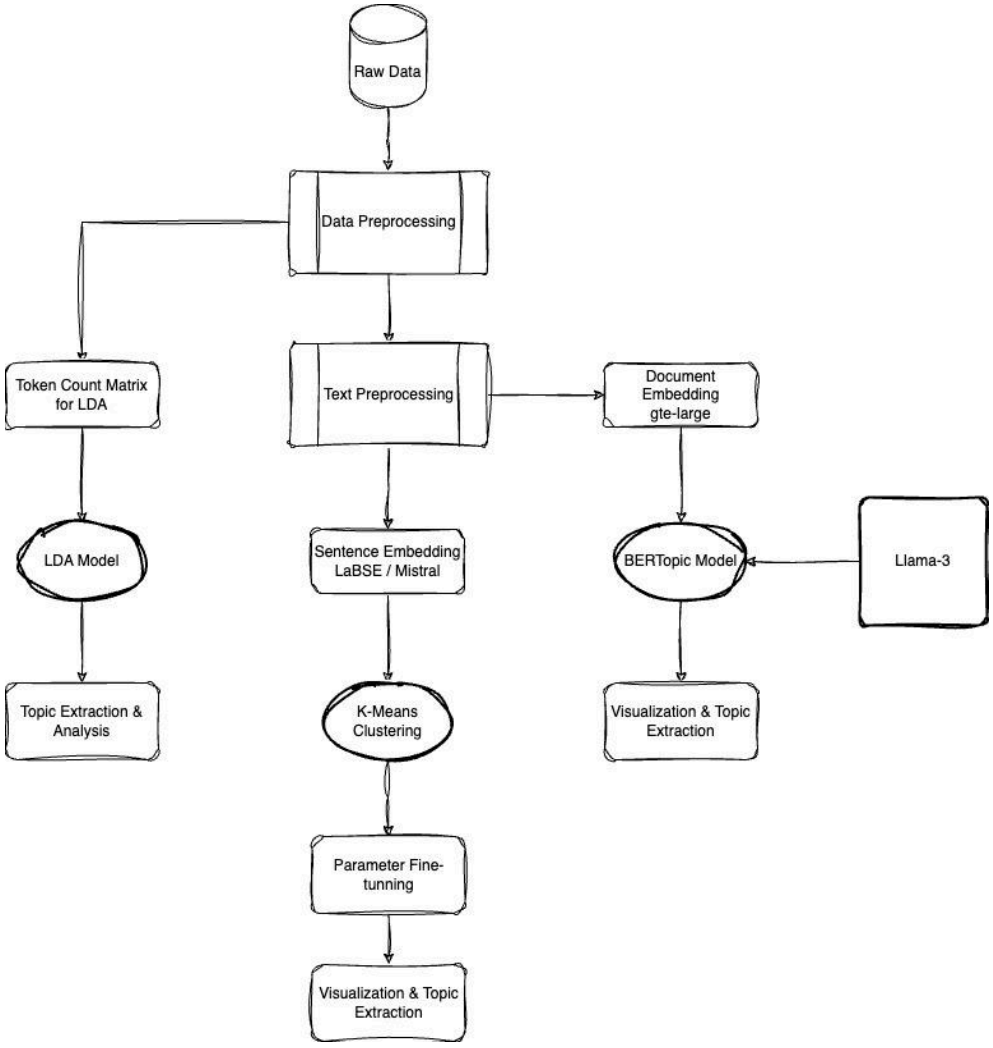

Fig. 1 - Workflow diagram of the three approaches we used in this project.

1. Topic Modeling using Latent Dirichlet Allocation (LDA)

Data Preparation:

- **Preprocessing:**
  The raw data from the CAD call log entries was presented to us in an unnormalized tabular format (i.e., with each log entry recorded in multiple rows). We pivoted and unstacked the data to be in normal form (i.e., with each log entry recorded in exactly one row).

- **Vectorization:**
  We used `CountVectorizer` to convert the 911 call log entries (i.e., the text under the feature `Event_Remarks_Text`) into a matrix of token counts. The vectorizer was set to include terms that appeared in at least 10 documents but no

more than 20% of the documents. This helped the model focus on relevant terms while filtering out noise. We also included n-grams ranging from unigrams to 4-grams to capture more context.

For more information about `CountVectorizer` and its use cases, please see this *Medium* blog post: https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c

Model Training:

- **LDA Model:**
  We implemented the Latent Dirichlet Allocation (LDA) algorithm to discover latent topics in the call logs. The LDA model was initialized with 30 topics and trained using the online learning method over 100 iterations.

- **Topic Distribution Extraction:**
  After fitting the model, we extracted the **theta** matrix, which represents the topic distribution for each document. Topic distribution refers to how much each topic contributes to a document.

  For our use case, we had many 911 call log entries (with each call log being a document) and we wanted to determine what main topics they were about. The LDA model helped us to do this by saying, "Okay, let's imagine there are a number of topics that these documents could be about, like ‘assault,’ ‘domestic dispute,’ ‘robbery,’ etc., and each document is made up of a mixture of these topics."

  The topic distribution tells you, for a specific document, how much of each topic is present. For example, if you have a 911 call log about robbery, the topic distribution might show that 40% of the document is about robbery, 30% is about assault, and 30% is about harassment and threat.

  By analyzing these topic distributions across many documents, we could start to see patterns and understand what topics appear to be common or important in our collection of documents.

Topic Analysis:

- **Term Extraction:**
  From the **beta** matrix, which represents the term distribution for each topic, we

identified the top 15 terms contributing to each topic. Terms with a pseudo count of at least 5 were selected to ensure relevance. Term distribution refers to the likelihood of each word appearing in a particular topic.

Again, imagine we have a list of topics, like "assault," "domestic dispute," and "robbery." Each topic has a set of words or phrases that are likely to appear when that topic is being discussed. For example, in the "robbery" topic, you might expect words like "firearms," "weapons," and "bank."

The term distribution for a topic tells you how likely each word is to appear in that topic. So, for the "robbery" topic, the term distribution might show that the word "weapons" has a high likelihood of appearing, while the phrase "food poisoning" has a low likelihood.

By looking at these term distributions, we could begin to understand what words were most characteristic of each topic, helping us interpret and analyze the topics that the LDA model identified in our text data.

- **Elbow Plot:**
  To determine the optimal number of topics, we generated an elbow plot by calculating the **perplexity** of LDA models with varying numbers of topics (from 2 to 20). The optimal number of topics was identified at the point where the rate of decrease in perplexity slowed.

  Perplexity is a measure often used to evaluate the performance of language models. In the context of LDA, perplexity measures how well the model predicts a sample of text. A lower perplexity value indicates that the model is better at predicting the words in the text.

  When generating an elbow plot to determine the optimal number of topics in an LDA model, perplexity is used as a metric to compare different models with varying numbers of topics. The idea is to find a balance between having enough topics to capture the complexity of the data and avoiding overfitting (i.e., creating too many topics that don't generalize well to new and unseen data).

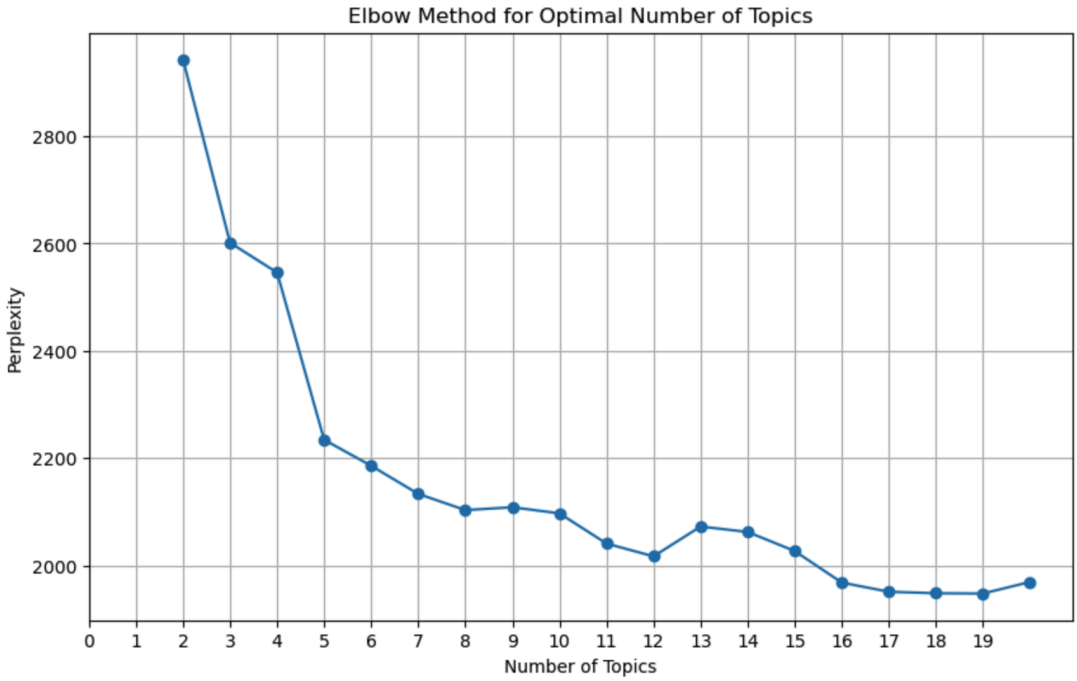

Fig. 2 - LDA Perplexity Elbow Plot

In this elbow plot, the x-axis represents the number of topics, and the y-axis represents the perplexity. The "elbow point" on the plot is where the rate of decrease in perplexity slows down significantly. This point is considered optimal because it indicates that adding more topics beyond this point does not lead to significant improvement in the model's ability to predict the text. Therefore, the elbow point is often chosen as the optimal number of topics for the LDA model.

Here we picked 9 as the number of clusters to balance the interpretability and complexity of the model.

Topic Mapping:

- **Manual Mapping:**
  The identified topics were manually mapped to predefined categories based on the most prominent terms, facilitating easier interpretation and application of the findings.

2. Sentence Embedding & K-Means Clustering

Data Preparation:

- **Text Compilation:**
  We compiled detailed textual descriptions for each event log entry by

concatenating various fields such as Occurrence_Type, Occurrence_Report_Category, Priority, and Event_Remarks_Text. This holistic view of each log entry provided a richer context for analysis.

- **Acronym Expansion:**
  Acronyms within the texts were expanded using a predefined glossary to enhance clarity and consistency.

Embedding Generation:

- **Sentence Embeddings:**
  We tested 2 embedding models, namely
  1. **LaBSE** (Language-Agnostic BERT Sentence Embeddings), and
  2. **Mistral 7B instruct**.

  We used these models to generate dense vector representations (i.e., embeddings) for each compiled text. **LaBSE** is designed to produce high-quality embeddings for a wide range of languages and contexts. **Mistral 7B instruct** is also pre-trained using various sources of text data and has a higher embedding dimension (i.e., embeddings with a higher number of dimensions usually are more capable of picking up subtle semantic nuances).

Clustering:

- **K-means Algorithm:**
  We applied the K-means clustering algorithm to group the embeddings into clusters based on cosine similarity. Initially, we set the number of clusters to 30.

  Key concepts of K-means algorithm:

  K-means: Imagine a number of points on a piece of paper, but they are not organized in any particular way. K-means is a method to group these points into clusters based on their similarity. It works like this:

  1. First, you randomly place some points on the paper as cluster centers (i.e., imagine these as the centers of your groups).

  2. Then, for each point, you calculate which cluster center it is closest to.

  3. After that, you move the cluster centers to the average position of all the points assigned to that cluster.

4. You keep repeating these steps, moving the cluster centers and reassigning points to clusters, until the clusters stop changing much. At this point, the algorithm has "converged," and you have your clusters.

Cosine Similarity:
Imagine you have two lists of numbers. Cosine similarity is a way to measure how similar these lists are, but not in the usual sense of "how close are these numbers to each other?" Instead, it measures the angle between these lists if you imagine them as vectors (i.e., like arrows pointing in space, and remember, our embeddings are vectors).

- If the angle is 0 degrees, it means the lists are very similar (pointing in the same direction).
- If the angle is 90 degrees, it means the lists are completely different (pointing in perpendicular directions).
- The cosine similarity value ranges from -1 to 1, where 1 means the lists are exactly the same, 0 means they are orthogonal (i.e., completely unrelated), and -1 means they are exactly opposite.

K-Means is a way to group similar points, and cosine similarity is a way to measure how similar two sets of numbers are by looking at the angle between them.

Cosine Difference:
Instead of using the standard Euclidean distance as the inter-cluster-distance measure for K-Means, we used cosine distance (cosine distance = 1 - cosine similarity). Cosine distance is a more robust metric to measure the dissimilarity of embeddings semantically.

- **Elbow Method:**
  To determine the optimal number of clusters, we plotted the **inertia** for K values ranging from 1 to 20 and identified the elbow point, finalizing the number of clusters at 9.

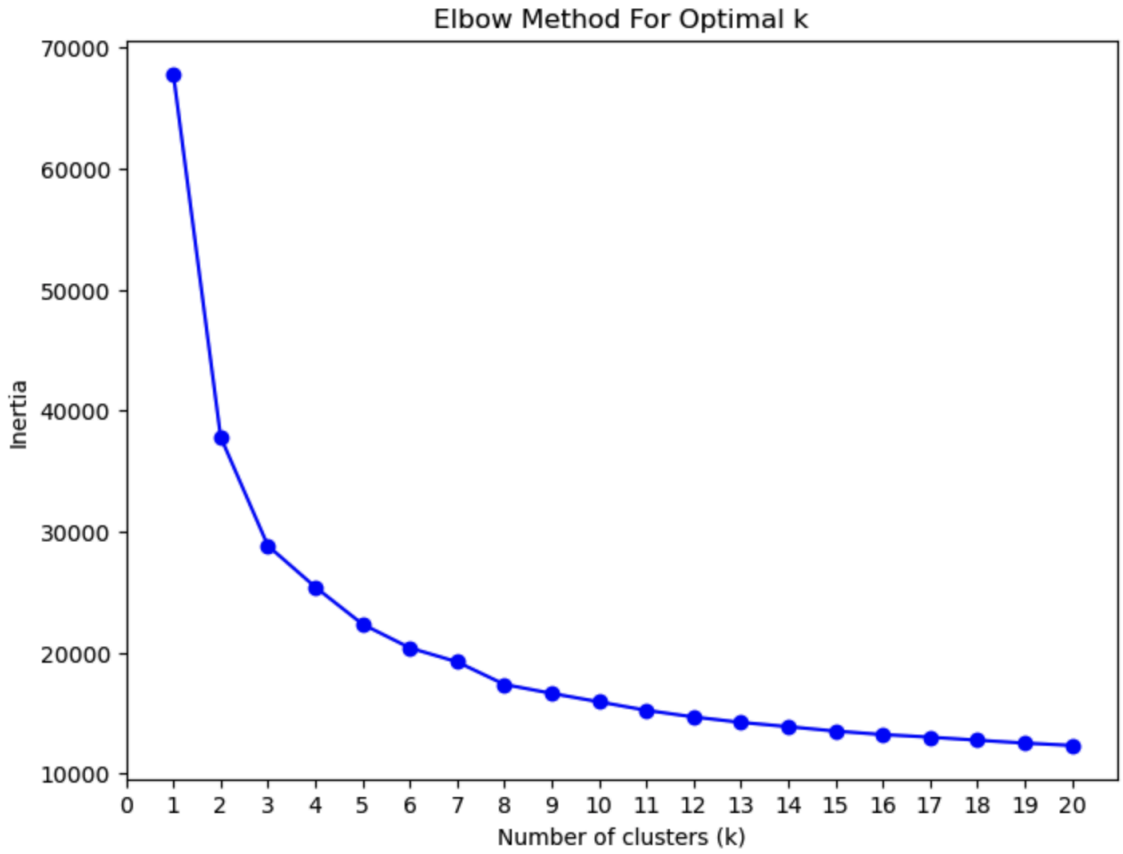

Fig. 3 - K-Means Inertia Elbow Plot

Inertia refers to the sum of squared distances between data points and their assigned cluster centroids. It is a measure of how tightly clustered the data points are within their clusters.

When using the elbow method to determine the optimal number of clusters for K-means, we plot the inertia values for different numbers of clusters. The idea is to find the point on the plot where adding more clusters does not lead to a significant decrease in inertia. This point is often referred to as the "elbow point" and is considered a good choice for the number of clusters because it represents a trade-off between having enough clusters to capture the data's structure and not having too many clusters that would overfit the data.

Visualization:

- **t-SNE:**
  To visualize the clustered embeddings, we used **t-distributed Stochastic**

Neighbor Embedding (t-SNE) to reduce the dimensionality to 2D. This helped in visually inspecting and validating the clustering results.

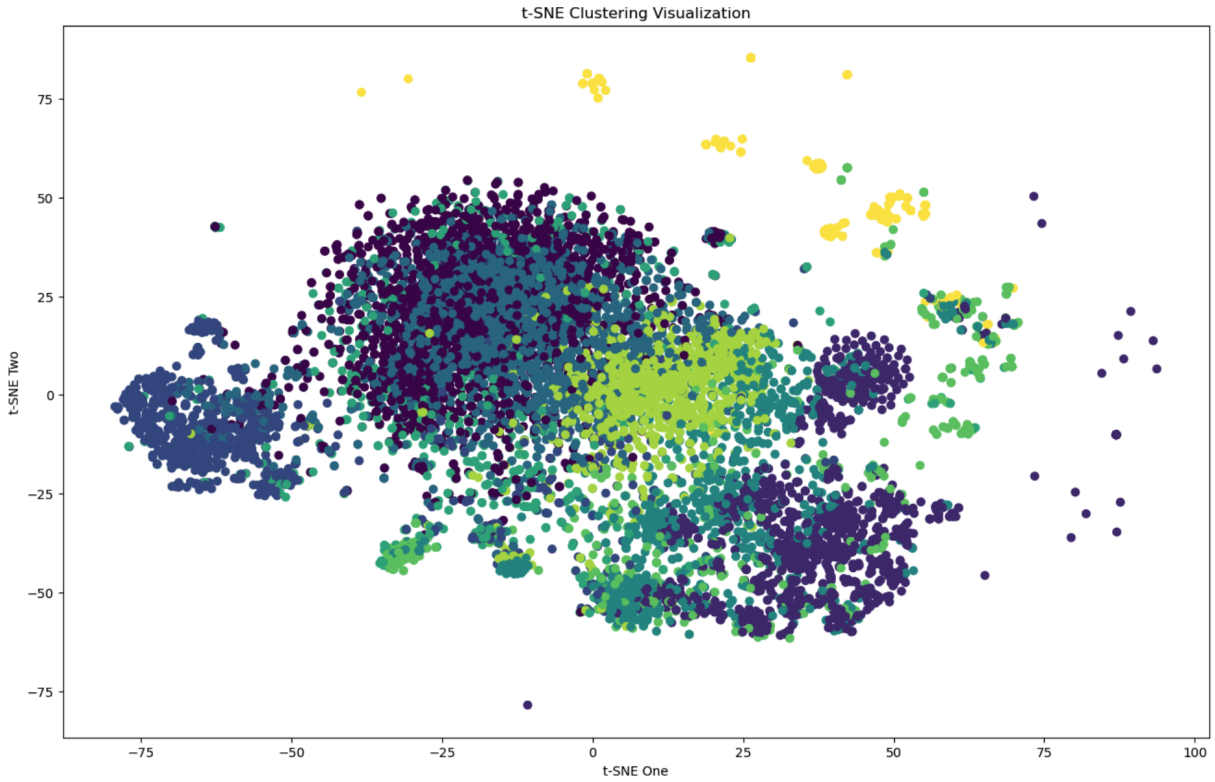

Fig. 4 - t-SNE plot of LaBSE embedding clusters

Imagine you have a lot of data points, each with many features. It can be hard to understand this complex data because it has so many dimensions. t-SNE helps by compressing this data down into just two or three dimensions, which we can easily see on a regular graph.

When it does this compression, it tries to keep similar points close together and dissimilar points far apart. So, if two points were close together in the original high-dimensional space, they will still be close together in the t-SNE visualization. This helps us see clusters of similar points and understand how they relate to each other.

3. Zero-shot Topic Modeling using BERTopic

Data Preparation:

The same compiled textual descriptions used for the K-Means clustering approach were employed here to maintain consistency.

Model Training:

- **BERTopic:**
  We utilized BERTopic, a topic modeling package that leverages pre-trained transformer models for generating embeddings and combines them with clustering techniques. BERTopic can operate in a zero-shot manner, meaning it does not require labeled data for training. We also leveraged BERTopic's new feature under its Representation Models module to use LLM (Large Language Models) to fine-tune the representation of clusters, allowing us to generate meaningful labels for topics. For the cluster representation fine-tuning, we used the open-sourced Llama-3 8B model, which has been quantized to 8 bits.

- **Embedding Model:**
  The `gte-large` model was used within BERTopic for generating embeddings since it is known for its high performance in the MTEB clustering task. This model is compact and captures the semantic nuances of the text, making it suitable for topic modeling.

  For more information about `gte-large` model, please see:
  https://huggingface.co/thenlper/gte-large

Topic Extraction Behind the Scenes:

BERTopic automatically generated topics based on the semantic structure of the text. The model identified coherent topics and assigned them to the corresponding documents with just a couple of lines of code.

```
from bertopic import BERTopic
from sklearn.datasets import fetch_20newsgroups

docs = fetch_20newsgroups(subset='all',  remove=('headers',
'footers', 'quotes'))['data']

topic_model = BERTopic()
topics, probs = topic_model.fit_transform(docs)
```

Code snippet from maartengr.github.io/BERTopic

However, with one function call, BERTopic handles 6 key topic-modeling steps. These steps are indicated in the diagram below. We will dive deeper into each step and briefly discuss our choice of implementation and their default settings.

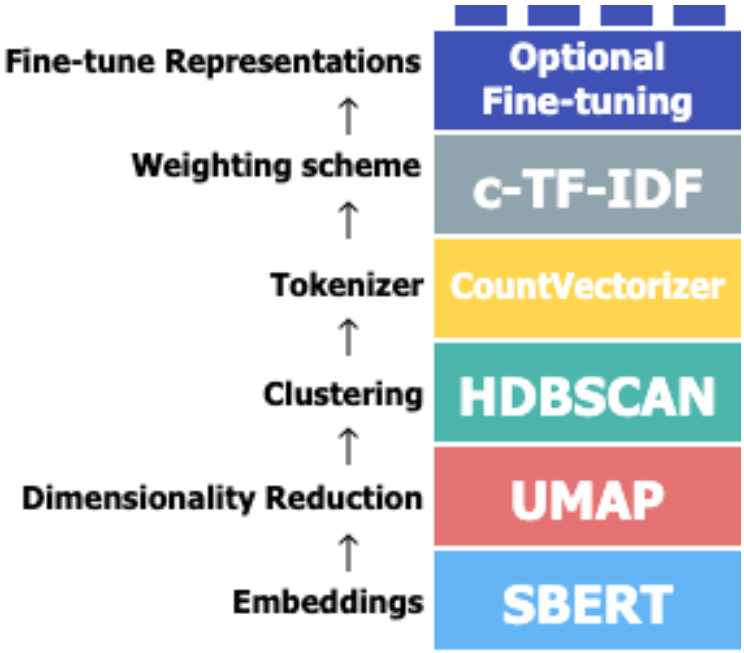

Diagram from maartengr.github.io/BERTopic/

Fig. 5 - BERTopic modularized essential steps to topic modeling

BERTopic operates through a sequence of 6 key modules:

1. Embeddings:
This module converts documents into dense vectors using pre-trained language models. For our project, we used the `thenlper/gte-large` model, which has been shown to perform well in clustering tasks. The default embedding model, `all-MiniLM-L6-v2`, also converts text into high-dimensional vectors that capture semantic meanings.

2. Dimensionality Reduction:
To make clustering more efficient, BERTopic reduces the dimensionality of the document embeddings. It offers various techniques like UMAP (Uniform Manifold Approximation and Projection) and PCA (Principal Component Analysis). UMAP is the default setting and our choice for this project. It is particularly effective because it preserves the local and global structure of the data, making clusters more distinguishable.

For more information about UMAP and PCA, please see these 2 *Medium* blog posts:

UMAP: https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

PCA:
https://medium.com/@noorulhudaajmal12/understanding-principal-component-analysis-pca-f831f0ce08c5

3. Clustering:
The reduced-dimensionality embeddings are clustered using the algorithm called HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise). HDBSCAN is preferred because it can identify clusters of varying shapes and sizes and handle noise effectively. It assigns each document to a cluster, or labels it as noise if it doesn't fit well into any cluster.

For more information about HDBSCAN, please see this *Medium* blog post:
https://towardsdatascience.com/a-gentle-introduction-to-hdbscan-and-density-based-clustering-5fd79329c1e8

4. Tokenizer:
The tokenizer splits documents into tokens, which are then used to generate topic representations. BERTopic can use a variety of tokenizers, such as whitespace, or more sophisticated tokenizers like CountVectorizer from scikit-learn, which we used in our implementation. The tokenizer's role is crucial for transforming text into a format that can be easily processed for keyword extraction.

5. Weighting Scheme:
To identify the most representative words for each topic, BERTopic uses a weighting scheme like class-based Term Frequency-Inverse Document Frequency (c-TF-IDF). This scheme calculates the importance of words or phrases in each topic by considering their frequency in the topic relative to their frequency in the entire corpus.

For more information about c-TF-IDF, please see this *Medium* blog post:
https://towardsdatascience.com/creating-a-class-based-tf-idf-with-scikit-learn-caea7b15b858

6. Representation Tuning:
This final step refines the topic representations using models like LLMs (Large Language Models). In our case, we used the 8-bit quantized Llama-3 8B model with a custom prompt to generate topic labels based on the most representative documents and keywords. This step enhances the interpretability and quality of the topics by

providing human-readable labels and summaries.

Replication of Methods

Detailed descriptions and scripts for each step are provided in the appendix to facilitate understanding and implementation.

Evaluation

Our model evaluations mostly rely on extrinsic measures and qualitative analysis. We initially planned to apply some intrinsic measures like Silhouette coefficient and Calinski-Harabasz Index. However, we soon realized that it is difficult to interpret the results of these metrics, especially when applying them on cluster outputs of different algorithms. Thus, intrinsic metrics were mainly used as a tool of fine-tuning individual models, rather than evaluation measures.

For **extrinsic evaluation**, we utilized some of the variables included in the given data: i.e., *Public_Generated_Event_Flag*, and *Initial_Dispatched_Event_Type_Code_Desc*. These variables represented some aspects of the nature of each emergency call, so good clusterings should have aligned with the value distributions of these variables.

We used mutual information as a metric, because it would measure how much information of the features are contained in the cluster labels.

	Sentence Embedding & K-Means Clustering	Zero-shot Topic Modeling using BERTopic
Public_Generated_Event_Flag	0.134	0.084
Initial_Dispatched_Event_Type_Code_Desc	0.104	0.149

Note that the mutual information score for the BERTopic model would suffer because of having a large number of clusters, despite the individual clusters aligning well with the features. This is an inherent problem of using mutual information to compare different models, so we had to rely on qualitative analysis to evaluate the model qualities.

For **qualitative analysis**, we used word clouds and other visualizations. We relied on the domain knowledge of the partners at the CPS in order to judge whether or not the clusters we were making were interpretable and reasonable. We will include a few word clouds here that are representative of the ones we showed to the partners at the CPS, as these played a key part in receiving feedback on the work we had done. Further discussions of these word clouds will be presented in the Analysis section below.

Fig. 6 - A word cloud of the key phrases for one of nine clusters made using LaBSE.

Analysis

As stated previously, we needed to rely on our partners at the CPS to qualitatively evaluate the clusters we made. The CPS provided positive feedback, expressing satisfaction with our clustering results. They indicated that we had made some insightful groupings of calls, revealing some types of calls which were being commonly reclassified as X99 Miscellaneous but which they had not previously identified. As our main task for this project was to investigate why X99 calls had increased in recent years, and we were apparently able to yield some insightful discoveries, our project can be considered an overall success.

Our final round of clustering using BERTopic was performed with a minimum cluster size of 25 events, resulting in a total of 76 clusters. Of 9,752 total events in the dataset, 3,580 events (36.7%) were found not to fit into any other clusters given the minimum cluster size. Because these calls could not be grouped with the others, they could be considered to be the most “truly miscellaneous” of the X99 calls provided to us.

As 76 clusters were too many to expect our partners to verify manually, we chose to consider and present only the 10 most-numerous clusters. The largest cluster had 588 calls associated with it, and Llama-3 labelled it as relating to “Vehicle Theft.” The following is a word cloud for this cluster:

Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

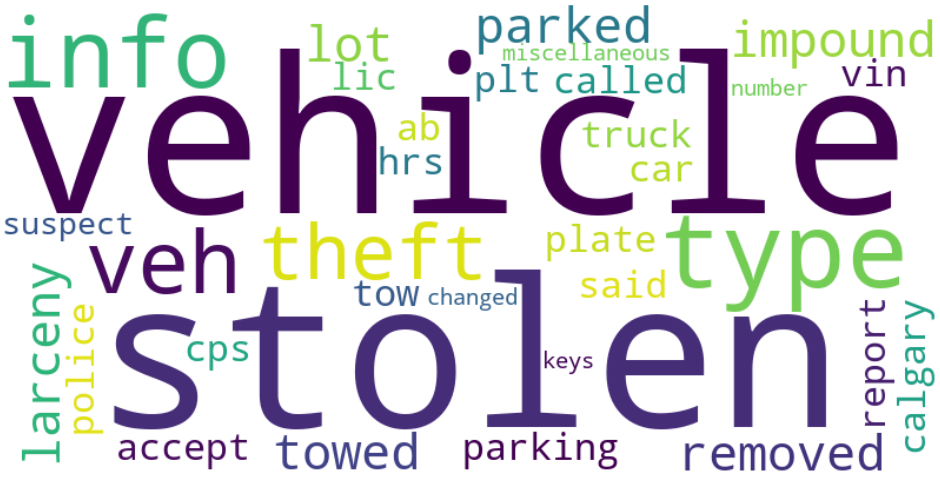

Fig. 7 - A word cloud for the cluster labeled "Vehicle Theft."

Here, we can see that the words in the word cloud do indeed appear to reflect a theme of "vehicle theft." The Llama-generated labels provided a way for the partners to rapidly parse through large amounts of clusters and be able to tell what the calls had in common at a glance. "Vehicle theft" is a somewhat fitting label, as all the data in this cluster does have to do with vehicles. However, not all events in this cluster are actual *thefts* (though some are). The Llama-generated labels are not always perfectly accurate, but they do seem to be a helpful tool for getting a general idea of what a cluster is about and saving time on manual evaluation.

There were some clusters in particular which our CPS partners stated were insightful, and brought their attention to a type of call that was being classified as X99 which they were not aware of. The following cluster contained 174 calls relating to 'paid duty':

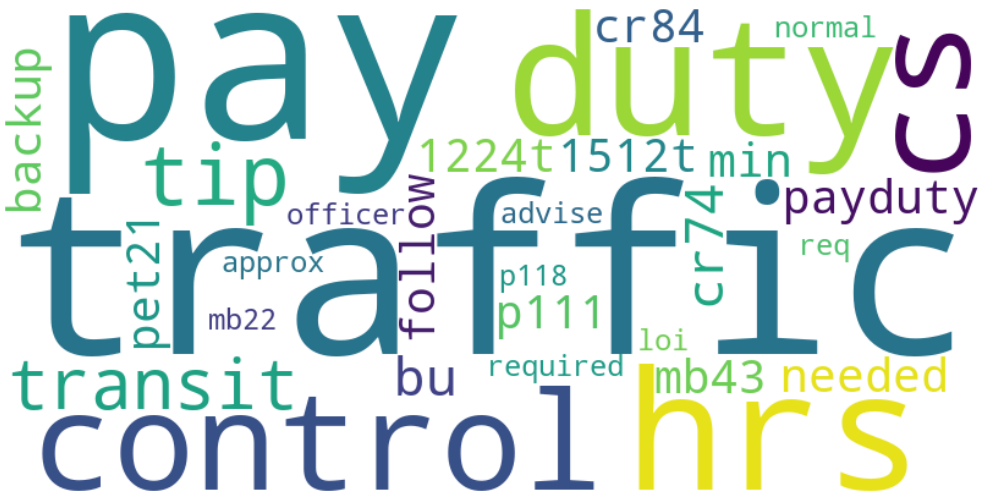

Fig. 8 - Cluster labeled "Law Enforcement Incident Reports - Officer Attendance."

Our partners at the CPS confirmed that this cluster appeared to be related to when off-duty police officers are hired on paid-duty, meaning they are paid to perform some task, such as traffic control, transit policing, or acting as backup security for events such as hockey games. It seems our model was able to identify a potential source of the increase in miscellaneous calls: officers classifying paid-duty requests as X99 calls. Using a greater number of clusters with BERTopic offered us more fine-grained clusters such as this one. These more specific clusters appear to be a more identifiably distinct source of X99 calls, and are therefore hopefully highly actionable in terms of the CPS being able to improve the classification of 911 calls.

Our BERTopic model did seem to have some slight shortcomings in dividing clusters too finely, in that it would sometimes separate groups of calls which reasonably ought to have been combined. For example, take the following two clusters, which had 169 calls and 161 calls associated with them, respectively:

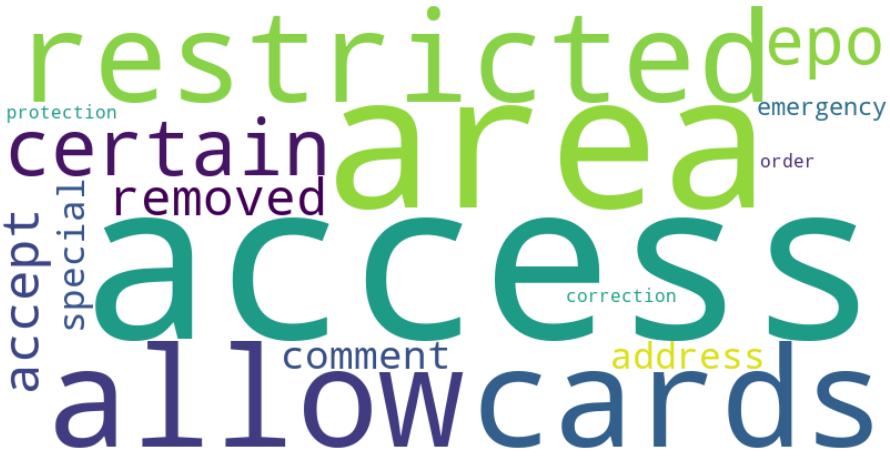

Fig. 9 - Cluster labeled “Emergency Protection Orders (EPOs) in Restricted Access Areas.”

Fig. 10 - Cluster labeled “Police information reports.”

As can be noticed from a cursory glance at both word clouds, the clusters are incredibly similar in their content, both relating (in the CPS's estimation) to violations of an Emergency Protection Order (EPO). Looking at the data itself, all the calls in the first cluster can be observed to follow the same strict format, all beginning with "*Accept Advised Event: SPECIAL ADDRESS COMMENT: This is a restricted access area and only certain access cards allow access to this area. EPO - [Name Removed]*." The calls in the second cluster also relate to violations of EPOs, but none of the calls follow this format. It seems that the model is placing too much importance on the identical wording in the first cluster, rather than the underlying semantic content of the calls, and that these clusters should actually be combined. This type of error once again highlights how important it was to qualitatively evaluate the data and use domain knowledge to make judgements during this project, and ultimately these errors were not a large issue for our ability to provide useful findings to the CPS regarding why X99 calls have increased.

Future Work

Shortcomings and Limitations

1. **Data Quality and Quantity:**
    - Limited Data Volume:

      The current dataset only contains 3 months worth of data and may not be comprehensive enough to capture all relevant topics, particularly those that are rare or emerging. A larger dataset would likely improve the robustness and coverage of the topic model.

    - Data Preprocessing:

      The preprocessing steps could be further refined to handle noise, such as spelling errors, slang, or incomplete sentences, which could affect the quality of the embeddings and, consequently, the topic clusters.

2. **Model Performance:**
    - Embedding Models:

      While we used advanced models like LaBSE, Mistral 7B, and thenlper/gte-large, there may be newer or more specialized models that could provide better embeddings for our specific use case.

    - Clustering Sensitivity:

      The performance of clustering algorithms like HDBSCAN can be sensitive to parameter settings. Further hyperparameter tuning and experimentation

with different clustering algorithms could yield more stable and meaningful clusters.

3. **Topic Interpretability:**
    - Representation Models:

      Although the Llama-based representation model improves topic labeling, the interpretability of topics can still be subjective. More sophisticated or customized prompts might enhance the quality of the generated labels.

    - Keyword Extraction:

      The quality of extracted keywords can significantly impact topic interpretability. Exploring alternative weighting schemes or hybrid approaches combining multiple techniques might yield better results.

Future Directions

1. **Data Expansion and Enrichment:**
    - Collecting More Data:

      To improve model performance, collecting additional data from diverse sources will be crucial. This includes both expanding the volume of data and ensuring it covers a wider range of topics.

    - Annotation and Labeling:

      Developing a labeled dataset for topic modeling could help evaluate and fine-tune the model more effectively. Leveraging domain experts could provide valuable labeled data.

2. **Model Enhancements:**
    - Exploring New Embedding Models:

      As new embedding models are developed, particularly those fine-tuned for specific domains (such as policing), integrating them into our pipeline could enhance topic detection and clustering.

    - Advanced Clustering Techniques:

      Experimenting with other clustering algorithms, such as DBSCAN, Gaussian Mixture Models, or even hierarchical clustering, may uncover better-suited methods for our data characteristics.

3. **Improving Interpretability:**
    - Custom Prompt Engineering:

Refining prompts for the Llama-3 model or integrating other LLMs like GPT-4 with specialized prompts could yield more accurate and informative topic labels.

- User Feedback Loop:

  Implementing a system to collect user feedback on the generated topics and labels could provide continuous improvement. This feedback could be used to adjust the model and improve its interpretability and relevance over time.

4. **Integration and Deployment:**
   - Real-time Topic Detection:

     Developing a system for real-time topic detection and analysis could be highly valuable for applications like social media monitoring or customer feedback analysis.

   - Visualization Tools:

     Creating interactive visualization tools to explore and interpret the topic models could make the results more accessible and actionable for stakeholders.

By addressing these shortcomings and exploring the suggested future directions, we would have the potential to significantly enhance the quality, applicability, and user-friendliness of our topic modeling approach.

References

Berba, P. (2020, July 8). A gentle introduction to HDBSCAN and density-based clustering. Medium. https://towardsdatascience.com/a-gentle-introduction-to-hdbscan-and-density-based-clustering-5fd79329c1e8

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2020). Language-agnostic BERT sentence embedding. arXiv. https://doi.org/10.48550/arXiv.2007.01852

Grootendorst, M. (2020, October 19). Creating a class-based TF-IDF with Scikit-Learn. Medium. https://towardsdatascience.com/creating-a-class-based-tf-idf-with-scikit-learn-caea7b15b858

Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv. https://doi.org/10.48550/arXiv.2203.05794

Jain, P. (2021, May 4). Basics of CountVectorizer. Medium. https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., & Zhang, M. (2023). Towards general text embeddings with multi-stage contrastive learning. arXiv. https://doi.org/10.48550/arXiv.2308.03281

Oskolkov, N. (2019, October 3). How exactly UMAP works. Medium. https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

Swati, H. (2023, September 25). Understanding principal component analysis (PCA). Medium. https://medium.com/@noorulhudaajmal12/understanding-principal-component-analysis-pca-f831f0ce08c5

Appendix

Timeline

Week 1 (May 6 - May 12):

As we received our data later than anticipated, and only received the smaller sample set, we took the remainder of Week 1 to perform EDA and gain a better understanding of the structure of the data. Additionally, as the text field required considerable domain knowledge to interpret, we devised questions to ask the CPS including the meanings of acronyms, police codes, etc.
We also collaboratively completed the project plan.

Week 2 (May 13 - May 19):

In Week 2, we devised a preprocessing strategy and implemented our first clustering models as a baseline.
Week 2 Contribution table:

Name	Contributions
Hayden	- Experimented with LDA topic modeling
Jingyi	- Experimented with n-gram agglomerative clustering

Jarrett	- Experimented with n-gram agglomerative clustering
Yuxi	- Data preprocessing

Week 3 (May 20 - May 26):

In Week 3, we explored some more clustering strategies, including using sentence embeddings.

Week 3 Contribution table:

Name	Contributions
Hayden	- Experimented with embeddings & K-Means clustering model
Jingyi	- EDA
Jarrett	- Researched alternative clustering methods - Generated word clouds for sharing with partners
Yuxi	- More EDA

Week 4 (May 27 - June 2):

Early on in Week 4, we met with the capstone group working with BCI, as our mentor Jungyeul felt that the projects were similar and that we should exchange ideas and methods we had tried. They introduced us to BERTopic, which we spent the remainder of the week researching. We also presented the clusters we had created using LaBSE to the CPS in order to receive feedback.

Week 4 Contribution table:

Name	Contributions
Hayden	- Researched topic modeling methods
Jingyi	
Jarrett	- Created slides for sharing clustering results with the CPS + cleaned spreadsheet of clustered data - Presented clustering results in weekly partner meeting, answered follow-up questions
Yuxi	(Nothing) :(

Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

Week 5 (June 3 - June 9):

In Week 5, we continued to experiment with BERTopic, and completed the report skeleton and blog post. The CPS had supplied us with a bit of feedback by this point in the form of some stop words, but had not yet weighed in much on the validity of our clusters themselves. As such, we chose to wait for more feedback before showing any BERTopic clusters to the CPS.

Week 5 Contribution table:

Name	Contributions
Hayden	- Contributed to report skeleton - Contributed to blog post
Jingyi	- Experimented with BERTopic
Jarrett	- Contributed to blog post - Contributed to report skeleton - Experimented with BERTopic
Yuxi	- Experimented with BERTopic

Week 6 (June 10 - June 16):

In Week 6, we incorporated the feedback provided by the CPS into creating some new clusters using BERTopic, as well as generating topic labels using Llama-3. We then presented these clusters to the CPS. Besides this, we also wrote the report draft.

Week 6 Contribution table:

Name	Contributions
Hayden	- Experimented with BERTopic Model - Contributed to Methods and Future work sections of Report Draft
Jingyi	- Contributed to summary section of Report Draft
Jarrett	- Created word clouds for sharing with partners - Created slides for sharing new clustering results with the CPS + updated cleaned spreadsheet of clustered data - Presented clustering results in weekly partner meeting

	- Contributed to Data and Analysis sections of Report Draft + general editing
Yuxi	- Working on extrinsic evaluation metrics - Evaluation section of Report Draft

Week 7 (June 17 - June 23):

Our final presentation for the partners at the CPS is scheduled for Tuesday, June 18th, at 9:00 am PST. We have been instructed to present for 15-20 minutes about the project, incorporating both technical details and explanations for lay people. Afterwards, we will have approximately 10 minutes to take questions.

We will also be completing the final draft of this report, which is due Wednesday, June 19th at 11:59 pm. As editing the final draft likely should not take as much time as writing the rough draft did, we can also likely spend a fair amount of time preparing our final deliverables this week.

Week 8 (June 23 - June 25):
Our final deliverables are due on June 25th at 11:59 pm, and our final presentation for the cohort is scheduled for June 28th.

Replicating the Steps

For those looking to replicate our methodology, here are the key steps:

1. **Data Preparation:**

   Depending on if the raw data is in the same format, if so, you can use `read_encrypted_data()` and `reformat_data()` from `preprocess.py` to attain the normalized data.

```
def read_encrypted_data(passwd, path_to_data):
    decrypted_workbook = io.BytesIO()
    with open(path_to_data, 'rb') as file:
        office_file = msoffcrypto.OfficeFile(file)
        office_file.load_key(password=passwd)
```

```
        office_file.decrypt(decrypted_workbook)

    return pd.read_excel(decrypted_workbook)
```

```
def reformat_data(data):
    df = data.copy()
    df["Event_Remarks_Text"] = df["Event_Remarks_Text"].astype(str)
    df = df.sort_values(by=["Event_Anonymizer",
"Event_Remarks_Created_Timestamp", "Remarks_Line_Order"],
ascending=True)
    event_group = df.groupby(["Event_Anonymizer"], as_index=False,
sort=False)

    agg_funcs = {
        col: 'first' for col in ["Grouped_Event_Type_Code_Desc",
"Grouped_Event_Subtype_Code_Desc",
                                 "Occurrence_Type",
"Occurrence_Type_UCR_Category", "Occurrence_Report_Category",
                                 "Priority",
"Public_Generated_Event_Flag", "Event_Attended_Flag"]
    }
    agg_funcs["Event_Remarks_Text"] = lambda x: "\n".join(x)

    return event_group.agg(agg_funcs)
```

Call `hybrid_expand_acronyms` function to expand the acronyms in the log entries.

```
def hybrid_expand_acronyms(text, glossary=glossary):
    for acronym, full_term in glossary.items():
        # Use word boundaries to replace only whole words
        text = re.sub(rf'\b{acronym}\b', f'{acronym} ({full_term})',
text)
    return text
```

Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

The variable glossary is a dictionary that contains all frequently used acronyms in the log entries.

```
glossary = {
    "CO": "Complainant",
    "VI": "Victim",
    "OFF": "Offender",
}
```

Call `compile_text` function to consolidate other useful features along with the log entries into a single text column.

```
def compile_text(x):
    text = (
        f"Description of the behavior or criminal offense:
{x['Occurrence_Type']}. \n"
        f"Broadest level of categorization:
{x['Occurrence_Type_UCR_Category']}. \n"
        f"Secondary level categorization:
{x['Occurrence_Report_Category']}. \n"
        f"The priority level assigned to the call by the ECO (911
call taker): {x['Priority']}. \n"
        f"Was the call initiated by a member of the public?
{x['Public_Generated_Event_Flag']}. \n"
        f"Flag that indicates the call was attended in person by an
officer: {x['Event_Attended_Flag']}. \n"
        f"The log of the event:
{hybrid_expand_acronyms(x['Event_Remarks_Text'])}"
    )
    text = re.sub(r"\[redacted\]", "[MASK]", text,
flags=re.IGNORECASE)
    return text
```

Call `output_embedding` function to transform the column of compiled text into embeddings of your choice.

```
def output_embedding(txt, model):
    try:
        embd = model.encode(txt, device="mps")
        return embd
    except Exception as e:
        print(f"Error encoding text: {e}")
        return None
```

2. **LDA Topic Modeling:**

   Use CountVectorizer for vectorization, considering terms with specified document frequency constraints.

```
vectorizer = CountVectorizer(lowercase = True, min_df = 10, max_df =
0.2, stop_words="english", ngram_range=(1, 4))
Text_X = vectorizer.fit_transform(df["Event_Remarks_Text"])
```

   Initialize and train an LDA model with the desired number of topics.

```
lda = LatentDirichletAllocation(n_components=30,max_iter=100,
learning_method='online',random_state=0,verbose=2,evaluate_every=1)
```

   Extract and analyze the theta and beta matrices to identify topics and key terms.

```
theta = lda.fit_transform(Text_X.A)
beta = lda.components_/np.sum(lda.components_,axis=1,keepdims=True)
max_words = np.argsort(-beta, axis=1)[:,:15]
features = vectorizer.get_feature_names_out()

for i in range(lda.components_.shape[0]):
    print(f"topic: {i}")
    print(f"psuedo count: {lda.components_[i].sum()}")
    print([features[ind] for ind in max_words[i] if
lda.components_[i, ind] >= 5])
    print('\n')
```

Hayden Chiu, Jingyi Liao, Jarrett MacFarlane, Yuxi Wang
19 June, 2024

Fine-tuning the number of topics with the Perplexity elbow chart.

```
# the range of topic numbers to test
topic_numbers = range(2, 21, 1)

# Fit LDA models and compute perplexity
perplexities = []
best_lda = None
best_theta = None
best_perplexity = np.inf

for n in topic_numbers:
    lda = LatentDirichletAllocation(n_components=n,
max_iter=100,learning_method='online',random_state=0,
evaluate_every=1)
    theta = lda.fit_transform(Text_X.A)
    perplexities.append(lda.perplexity(Text_X))
    print(f"Number of topics: {n}, Perplexity:
{lda.perplexity(Text_X)}")

    # pocket the best lda
    if lda.perplexity(Text_X) < best_perplexity:
        best_lda = lda
        best_theta = theta
        best_perplexity = lda.perplexity(Text_X)
```

```
# Visualize the elbow plot
plt.figure(figsize=(10, 6))
plt.plot(topic_numbers, perplexities, marker='o')
plt.xlabel('Number of Topics')
plt.ylabel('Perplexity')
plt.xticks(np.arange(0, 20, step=1))
plt.title('Elbow Method for Optimal Number of Topics')
plt.grid()
plt.show()
```

3. **Sentence Embedding & K-Means Clustering:**

   Compile detailed textual descriptions for each document and generate sentence embeddings using the LaBSE model.

```
# Load sentence transformers
LaBSE_model = SentenceTransformer("sentence-transformers/LaBSE")
```

```
# Apply the preprocessing function to the dataframe
df['LaBSE_text_embedding'] = df.apply(lambda x:
output_embedding(compile_text(x), LaBSE_model), axis=1)
```

Apply K-Means algorithm

```
# Convert the embeddings to a suitable format for clustering
embeddings = df['LaBSE_text_embedding'].tolist()

embeddings_array = np.array(embeddings)

# Use KMeans with cosine distance
kmeans = KMeans(n_clusters=30, random_state=0)

# KMeans doesn't directly support cosine similarity, so we convert
cosine distance to Euclidean distance
cosine_sim_matrix = 1 - pairwise_distances(embeddings_array,
metric='cosine')
kmeans.fit(cosine_sim_matrix)

# Assign clusters back to the dataframe
df['LaBSE_cluster'] = kmeans.labels_
```

Use the elbow method to determine the optimal number of clusters.

```
# Elbow method to find the optimal number of clusters
inertia = []
K = range(1, 21)
```

```
for k in K:
    kmeans = KMeans(n_clusters=k, n_init="auto", random_state=0)
    # KMeans doesn't directly support cosine similarity, so we
convert cosine distance to Euclidean distance
    cosine_sim_matrix = 1 - pairwise_distances(embeddings_array,
metric='cosine')
    kmeans.fit(cosine_sim_matrix)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve
plt.figure(figsize=(8, 6))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.xticks(np.arange(0, 21, step=1))
plt.show()
```

Visualize the results using t-SNE.

```
# let’s say we have 9 clusters
kmeans = KMeans(n_clusters=9, random_state=0)

kmeans.fit(cosine_sim_matrix)

# Assign clusters back to the dataframe
df['LaBSE_cluster'] = kmeans.labels_

print(df[['Event_Remarks_Text', 'LaBSE_cluster']])

# Visualization using t-SNE
tsne = TSNE(n_components=2, random_state=0)
tsne_result = tsne.fit_transform(embeddings_array)
df['tsne-one'] = tsne_result[:,0]
df['tsne-two'] = tsne_result[:,1]
```

```
plt.figure(figsize=(16,10))
plt.scatter(df['tsne-one'], df['tsne-two'], c=df['LaBSE_cluster'],
cmap='viridis')
plt.xlabel('t-SNE One')
plt.ylabel('t-SNE Two')
plt.title('t-SNE Clustering Visualization')
plt.show()
```

4. **Zero-shot Topic Modeling with BERTopic:**

   Use the same compiled text descriptions.

```
docs = df.progress_apply(lambda x: compile_text(x),axis=1)
```

Instantiate a `zeroshot_topic_list` for zero-shot topic modeling.

```
# We define a number of topics that we know are likely in the
documents
zeroshot_topic_list = preprocess.get_type_codes()
zeroshot_topic_list.remove("911")
zeroshot_topic_list.remove("MAJOR")
zeroshot_topic_list = list(zeroshot_topic_list)
```

Apply BERTopic with the ` thenlper/gte-large` model to generate and analyze topics.

```
# Use llama.cpp to load in a 8-bit quantized Llama 3 8B
# and truncate each representative document to 1024 words

model_path = "PATH/to/your/LLM/model"

prompt =
"""<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful, respectful and honest assistant for labeling
topics.<|eot_id|><|start_header_id|>user<|end_header_id|>
```

```
I have a topic that contains the following documents:
[DOCUMENTS]

The topic is described by the following keywords: '[KEYWORDS]'.

Based on the information about the topic above, please create a short
label of this topic. Make sure you to only return the label and
nothing more.<|eot_id|><|start_header_id|>assistant<|end_header_id|>
"""

# These parameters may be subject to change depending on your
computer GPU
n_gpu_layers = -1
n_batch = 512
n_ctx=8192

llm = Llama(
    model_path= model_path + "Meta-Llama-3-8B-Instruct.Q8_0.gguf",  #
Download the model file first
    n_ctx=n_ctx,  # The max sequence length to use - note that longer
sequence lengths require much more resources
    n_threads=8,            # The number of CPU threads to use,
tailor to your system and the resulting performance
    n_gpu_layers=n_gpu_layers, # The number of layers to offload to
GPU, if you have GPU acceleration available
    n_batch=n_batch,
    f16_kv=True,
    chat_format="llama-3",
    verbose=True, #change to True if you want to investigate the logs
)

vectorizer_model = CountVectorizer(ngram_range=(1, 3),
stop_words="english")


main_representation =  LlamaCPP(
    llm,
    tokenizer='whitespace',
    doc_length = 1024,
```

```
    prompt = prompt
)

# Additional ways of representing a topic
aspect_model_1 = KeyBERTInspired(top_n_words=50)


representation_model = {
    "Main": main_representation,
   "Aspect1":  aspect_model_1,
}

# We fit our model using the zero-shot topics
# and we define a minimum similarity. For each document,
# if the similarity does not exceed that value, it will be used
# for clustering instead.
topic_model_gte_large = BERTopic(
    embedding_model="thenlper/gte-large",
    min_topic_size=25,
    zeroshot_topic_list=zeroshot_topic_list,
    zeroshot_min_similarity=.85,
    representation_model=representation_model,
    vectorizer_model = vectorizer_model,
    verbose=True,
    calculate_probabilities=True,
    top_n_words = 50
)
topics, probs = topic_model_gte_large.fit_transform(docs)
```

If you want to inspect the topic information from the model

```
topic_model_gte_large.get_topic_info()
```

To display the count of each topic

```
topic_model_gte_large.get_topic_freq()
```

To save the model

```
# save BERTopic model
topic_model_gte_large.save("data/tuned_gte_large_model",
serialization="safetensors")
```

To load a model

```
#load BERTopic model
a_topic_model = BERTopic.load("data/tuned_gte_large_model")
```

To generate a word cloud on each topic

```
def create_wordcloud(model, topic):
    text = {word: value for word, value in
model.get_topic(topic,full=True)["Aspect1"]}
    wc = WordCloud(background_color="white", max_words=1000)
    wc.generate_from_frequencies(text)
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.show()

# Show word cloud
create_wordcloud(topic_model_gte_large, topic=7)
```

By following these steps, the partner company can replicate our methodology and adapt it for future analyses.