Miles Vollmer, Marcus Roldan
CS4120 - Natural Language Processing
18 April 2024

Text Classification for Bike Lane Obstruction Detection

Abstract:
This project develops an unsupervised text classification system to enhance the detection and reporting of bike lane obstructions using public 311 service requests. By applying the Lbl2Vec model, a variant of Word2Vec, this study classifies unlabeled 311 report text descriptions into multiple classes based on predefined labels and keywords. This approach aims to improve urban planning and parking enforcement response strategies by providing actionable insights into recurrent obstructions, ultimately enhancing the safety and usability of city bike lanes. Through the Lbl2Vec model, a Silhouette Score of 0.70 was achieved. The model can label an unseen corpus of 311Illegal Parking Service Request descriptions, providing a basis for later downstream tools to enhance the information available to urban planners examining the safety and efficiency of Boston's bicycle network.

Introduction:

In urban areas like Boston, bike lanes and other infrastructure are frequently obstructed by various impediments, creating challenges for cyclists and city planners alike. To address this issue, this project utilizes natural language processing techniques to organize and categorize the extensive but uncategorized and unlabeled data collected through 311 Illegal Parking Service Requests. By applying the Lbl2Vec model, an unsupervised text classification approach, raw JSON Service Request objects are collected, pre-processed, and through the generation of word, label, and keyword embeddings by Lbl2Vec labels are assigned to text descriptions. These descriptions are associated with full Illegal Parking Service Requests which contain data about the location, date, and time, as well as attached photos of the incident. This methodology not only streamlines data analysis but also aids in identifying recurrent obstruction hotspots, thereby providing urban planners with a tool to enhance urban mobility and safety for cyclists and pedestrians alike.

Methodology:

Data Collection and Preparation

Boston and many other cities provide public API endpoints following the Open311 protocol. Through this endpoint, requests can be made for 311 Service Requests, with optional parameters for request type, date, and request status (active, opened, completed). Through the use of a Python script, HTTP requests are made to gather all available Illegal Parking Service Requests.

The requests are returned in the form of arrays of JSON objects, one for each service request. These object contain fields such as the request's unique ID, the date, time, and location of the request, any photos submitted with the service request, and most importantly, the free-form text description of the request.

Another Python script is utilized to prepare the gathered service request data for preprocessing, by extracting the description and ID of the request. Once this data is prepared, preprocessing on the descriptions is executed. The descriptions are cleaned, removing special characters and accents, as well as removal of standard NLP stop-words (using the NLTK stop-words English list), as well as the removal of a custom stop-word list computed from the data itself.

Data Pre-processing

The custom stop-word list consists of the words with a count which exceeds ten percent of the number of total documents in the corpus. For example, if the corpus consisted of 10,000 documents, words which appear more than 1,000 times are added to the custom stop-list. The reason for the removal of these words is the high similarity in context between the documents. All documents are related to illegal parking, meaning the top words are 'parked', 'parking', or 'car', and do not contribute much to the classification. Removing these words increases the contextual differences between the documents, aiding in classification.

One key stipulation, however, is that any word removed from descriptions cannot be one of the pre-defined keywords. The Lbl2Vec model takes in a set of keywords which correspond to the labels it assigns to documents. For example, the 'bike lane' label has the keyword 'bike'. These keywords are embedded jointly with document and word embeddings when the Lbl2Vec model is trained on the data.

Once the protected stop-word list (NLTK and high-frequency, with keywords removed) is defined, the data is preprocessed using gensim.utils simple_preprocess() method, tokenizing at the word level, with a minimum length of 2 and a maximum length of 15. Accents and special characters are removed and standardized, along with converting all text to lowercase. After tokenization, each document is given an integer tag which corresponds to its 311 Service Request ID. These tags are used by the Lbl2Vec model to keep track of each document, its vector, and predicted label. Finally, the data was split into training, validation, and testing subsets with a 70/15/15 split, respectively.

Model Training

The Lbl2Vec has various hyperparameters which can be adjusted to yield models tuned for different use cases and datasets. These parameters include a label-keyword list which is vectorized along with the corpus to calculate the best labels for each document, a similarity_threshold for the cosine similarity between a document vector and a label vector, as well as more standard Word2Vec model hyperparameters such as vector_size, min_count, and number of epochs.

Unfortunately, gensim does not offer any hyperparameter search algorithms, so a naive search of the hyperparameter space was conducted. Because of the nature of the hyperparameters (some being continuous variables) and the time needed to train and test the Lbl2Vec models, a narrow hyperparameter space was defined using intuition across multiple rounds.

The first round used the following hyperparameter space:

- window: [5, 20] (discrete)
- negative: [3,7] (discrete)
- similarity_threshold: [0.5, 0.8] (continuous, step= 0.05)

The best hyperparameters from this first round of tuning were as follows:

- window: 10
- negative: 5
- similarity_threshold: 0.5

The second round of hyperparameter tuning used the following space:

- window: 10
- negative: [3,7] (discrete)
- similarity_threshold: [0.3, 0.5] (continuous, step=0.05)

The resulting best hyperparameters from the second round of tuning were as follows and resulted in a silhouette score of 0.5437:

- window: 10
- negative: 5
- similarity_threshold: 0.3

Data:

Data Source

The dataset for this project was obtained from the municipal 311 public service request system (Boston Open311 API), which collects non-emergency reports from city residents. These reports are formatted as JSON objects, with fields indicating the date, time, and location of the report, as well as a text description and any photos submitted alongside the report. Using the HTTP Get request parameters, only reports under the Illegal Parking category were gathered.

Data Characteristics

The dataset contains ~10,000 Illegal Parking Service requests over the last 90 days. The Boston 311 API only allows requests to be gathered to this point. The corpus is compiled from the text descriptions from each request. These descriptions describe various illegal parking issues, from blocked bike infrastructure to double-parking. Below is a sample of two JSON service request objects and their fields.


```
},
{
    "service_request_id": "101005335934",
    "status": "open",
    "service_name": "Illegal Parking",
    "service_code": "Transportation - Traffic Division:Enforcement & Abandoned Vehicles:Parking Enforcement",
    "description": "Car completely blocking turn in fire lane",
    "requested_datetime": "2024-03-02T23:30:00Z",
    "updated_datetime": "2024-03-02T23:32:29Z",
    "address": "Intersection Of Sheafe St And Snow Hill St, Boston, Ma",
    "lat": 42.36655656,
    "long": -71.05657373,
    "token": "44c753cd-f779-40ba-836a-05b40a73c720"
},
{
    "service_request_id": "101005335925",
    "status": "open",
    "service_name": "Illegal Parking",
    "service_code": "Transportation - Traffic Division:Enforcement & Abandoned Vehicles:Parking Enforcement",
    "description": "Double parking alone mass ave. No enforcement. Really? A two lane becomes one lane. Are you all sleeping?",
    "requested_datetime": "2024-03-02T23:00:00Z",
    "updated_datetime": "2024-03-02T23:02:36Z",
    "address": "400 Massachusetts Ave, Roxbury, Ma, 02118",
    "lat": 42.34142191,
    "long": -71.08272374,
    "token": "1486c94d-350f-4dcd-a854-a1cb7e8455c4"
},
```


Data Preparation and Pre-processing

The data preparation and pre-processing was conducted as described above in the *Methodology* section, and included standard practices such as special character and accent removal, stop-word removal, and tokenization. Custom stop-word removal of high-frequency words is explained in the *Methodology* section.

Data Structuring

After pre-processing and preparation, the final step before the documents are given as input to the Lbl2Vec model for training is to tag each document. gensim.models.Doc2Vec provides the TaggedDocument object for this purpose, consisting of a list of tokenized strings, followed by an integer tag for the document. The Lbl2Vec model treats these tags as direct integers, so the service request unique IDs cannot be used as they are too large in size. Thus, a hash function was implemented which minimizes the hash values while preventing collisions. In this case, a list of service request IDs was kept, with the index of the ID in the list being used as the hash and tag for the document. This system is not sustainable as the dataset grows, but functions to minimize the size and values of the document tags.

Results:
Because of the unlabeled nature of the dataset, traditional evaluation metrics such as Accuracy, Precision, Recall, or F1 do not apply. One way to evaluate the effectiveness of the model is through analysis of the clustering of the document and label embeddings. The Silhouette Score is a metric that measures the quality of clustering achieved by the model. It is calculated for each sample and is composed of two distances: a, the mean distance to the other samples in the same cluster (cohesion), and b, the mean distance to the samples in the nearest cluster that the sample is not part of (separation). The silhouette score for a single sample is then given by (b−a)/max(a,b).

Our model achieved a Silhouette Score of 0.70 on the test data, indicating both strong cluster definition and high similarity within these clusters. Cluster definition means on average documents within a cluster are closer to each other than to other clusters, while also suggesting a strong level of separation between the clusters. The high similarity within the clusters is a measure of the *cosine similarity* between documents within a cluster to those outside a cluster. This indicates cohesion within the clusters.

However, manual visual inspection of the classification produced by the document is less than ideal. A sample of descriptions from the test data are shown below, alongside their assigned label and the similarity they achieve with that label. There are obvious errors made by the model, and better results could be achieved using a rule based system to search for keywords and variants of keywords.

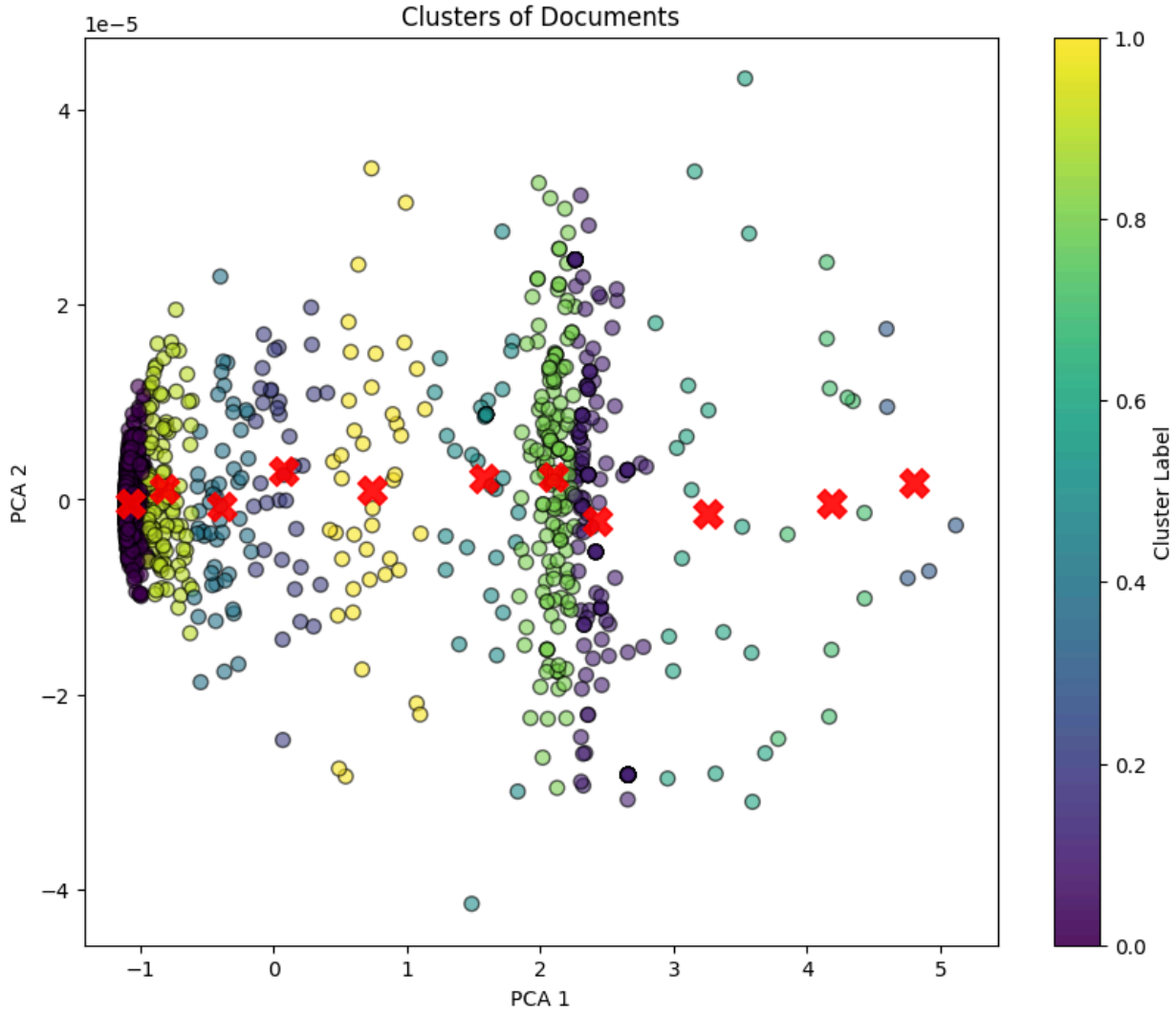
	doc_key	service_id	description	label	similarity
0	6746	101005410578	Car parked in the crosswalk.	visitor parking	-0.418184
1	6747	101005343758	Expired out of state please ticket	handicap	0.794816
2	6748	101005401099	Washington state	crosswalk	0.979388
3	6749	101005385928	Cars at school blocking access on Thomas Pk in...	double parking	0.987499
4	6750	101005352345	Non resident vehicle white Ford edge plate PS-...	double parking	0.996335
5	6751	101005403643	Red pick-up truck completely blocking crosswal...	visitor parking	0.986615
6	6752	101005336481	2hour parking limit and cars are parked all da...	driveway	0.997020
7	6753	101005334253	Car still parking on sidewalk	double parking	0.031528
8	6754	101005352188	-- auto translated (en) -- A citation was NOT ...	bike lane	0.999155
9	6755	101005372967	The constituent is a disabled woman who states...	double parking	0.998370

Below we can see a sample of the training documents, their assigned label, and their similarities to a subset of the labels. For each document, the variation of the similarities from a given document to each label is very low, often falling below 0.001. In other words, the documents are almost just as close to every possible label.

doc_key	most_similar_label	highest_similarity_score	bike lane	bus lane	resident parking	fire hydrant	sidewalk
0	visitor parking	0.613665	0.613659	0.613655	0.613663	0.613653	0.613655
1	double parking	0.869948	0.869941	0.869940	0.869941	0.869939	0.869938
2	bus lane	-0.235141	-0.235177	-0.235141	-0.235180	-0.235177	-0.235157
3	resident parking	0.997379	0.997378	0.997378	0.997379	0.997377	0.997377

These findings along with the strong silhouette score indicate that the overall clustering of the documents is quite tight, with the labels seemingly equidistant from every document. Future iterations should strive to improve the embeddings to increase the utility of the model.

A visual inspection of the clustering can also be achieved through plotting. First, PCA is applied to reduce the dimensionality to be more conducive to plotting. Then, K-means clustering is run on the reduced dimension space to visualize the clustering of the document embeddings. The resulting plot is shown below:



Discussion:

Multi-Label Classification

The current system categorizes each incident into a single category based on the highest cosine similarity score between document embeddings and label embeddings. While this method provides a straightforward classification scheme, it may oversimplify the complexity of urban issues. For instance, an incident where a vehicle is parked in a bike lane in front of a fire hydrant could be relevant to both 'bike lane obstruction' and 'blocked fire hydrant' categories. However, our current model would likely categorize it under only one class.

If the document and label embeddings can be improved in future iterations of the model, a threshold can be set, above which multiple labels can be selected as the classification of a document. This could be achieved by setting a hard threshold across documents (i.e. similarity >0.7 indicates a positive label), or by identifying outlier labels and using those as classification.

Advantages of Multi-Label Systems

Implementing a multi-label classification approach could offer several advantages, including more comprehensive categorization, improved response coordination, and enhancement of data analysis. More comprehensive categorization allows for a more nuanced understanding of incidents, as many urban issues are multifaceted and may fit into multiple categories simultaneously. By identifying all relevant categories for an incident, city services can coordinate more effectively. For example, the same report could alert both traffic control and emergency services. Multi-label classification can also provide richer data for analyzing patterns and trends in urban issues, leading to more informed decision-making in urban planning and resource allocation.

Future Research Directions

The largest obstacle to the effectiveness and utility of the model is the high similarity of the document corpus. In this case, the model must make classification decisions that are more nuanced than usual classifications. For example, the difference between a document describing a car parked blocking a crosswalk and a fire hydrant is much smaller than that between a document describing dog breeds and how to install Linux.

It is for this reason that high-frequency words that were not identified as keywords were removed from documents during pre-processing.

Once the model has improved to make more accurate classification decisions, potentially in a scheme which allows multiple labels to be applied, it can serve as the basis for further downstream tasks such as data analysis tools to aid urban planning decisions.

References:

The Lbl2Vec repository can be found at the following link: https://github.com/sebischair/Lbl2Vec
The paper produced by the creator of the Lbl2Vec model can be found using the following citation or link:

[1] T. Schopf, D. Braun, and F. Matthes, “Lbl2Vec: An embedding-based approach for unsupervised document retrieval on predefined topics,” Proceedings of the 17th International Conference on Web Information Systems and Technologies, Oct. 2022. doi:10.5220/0010710300003058
https://arxiv.org/abs/2210.06023

Appendix:

The file *full_pipeline.ipynb* is an interactive Jupyter Notebook which encompasses the entirety of the process used in the creation of this model. It begins with the *Data Collection* section, which outlines functions to collect the service request data from the Boston 311 API. The *Data Preparation* section continues the process of extracting and cataloging the relevant data from the collected service requests through a function to extract the service request ID and description. Next, the *Data Pre-processing* section outlines all steps taken for pre-processing, including keyword definition, frequency analysis, stop-word list curation and stop-word removal, as well as tokenization and tagging. The *Model Training* section shows the process of initializing and fitting the model, then predicting the labels of the training documents. The *Model Evaluation* section evaluates the model on the test documents, performs analysis through the Silhouette score, and creates visual inspection tools like the clustering plot and tables to manually inspect the classification of the documents.

The file *service_request_labeler.model* is a saved version of the Lbl2Vec model trained on the gathered corpus of service request descriptions. It is generated by the *full_pipeline.ipynb* file.

The file *service_requests_last_90_days.json* is a saved JSON array containing the raw JSON objects received by the Boston 311 API. It is generated by the *full_pipeline.ipynb* file.

The *requirements.txt* file outlines the libraries required to run the *full_pipeline.ipynb* file.