Development of the Crash Information Extraction, Analysis and Classification Tool (CIEACT)

Xiao Qin, Ph.D., PE[1]
Rohit Kate, Ph.D.[2]
Robert J. Schneider, Ph.D.[3]
Md Abu Sayed, MS[1]
Md Touhid Hossain MS[1]

1. Department of Civil and Environmental Engineering
2. Department of Computer Science
3. Department of Urban Planning
University of Wisconsin-Milwaukee

WisDOT ID FG-2020-UW-MILWA-05069 and FG-2021-UW-MILWA-05641

September 2022

RESEARCH & LIBRARY UNIT

WISCONSIN DOT

PUTTING RESEARCH TO WORK

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Development of the Crash Information Extraction, Analysis and Classification Tool (CIEACT)		**5. Report Date** September 2022 **6. Performing Organization Code**
7. Author(s) Xiao Qin, Rohit Kate, Robert. J. Schneider Md Abu Sayed, Md Touhid Hossain		**8. Performing Organization Report No.**
9. Performing Organization Name and Address Institute for Physical Infrastructure and Transportation (IPIT) University of Wisconsin-Milwaukee Milwaukee, WI 53201-0784		**10. Work Unit No.** **11. Contract or Grant No.**
12. Sponsoring Agency Name and Address Wisconsin Department of Transportation Research & Library Unit 4822 Madison Yards Way Room 911 Madison, WI 53705		**13. Type of Report and Period Covered** Final Report October 2021-September 2022 **14. Sponsoring Agency Code**

15. Supplementary Notes

If applicable, enter information not included elsewhere, such as translation of (or by), report supersedes, old edition number, alternate title (e.g. project name), or hypertext links to documents or related information.

16. Abstract

Crash narratives recorded by law enforcement officers contain valuable information that is not necessarily included in the structured data fields and attributes of a crash report. For the analysis and classification of crashes, particularly in cases involving misclassified or overlooked crashes, traffic safety practitioners and engineers depend on this crucial information within crash narratives. Extraction of such information is a challenging task, with safety practitioners mostly relying on manual work to sift through tens of thousands of narratives for relevant information. Not only is the manual review process labor intensive, but the review quality is inconsistent because it is subject to the reviewers' experience and judgement. Text mining and machine learning techniques are two alternative review methods that have proven to be efficient and effective in automatically extracting crucial information from crash narratives. The authors developed an online Crash Information Extraction, Analysis and Classification Tool (CIEACT) that would allow safety practitioners to effectively use these less time consuming and accurate techniques. The Noisy-OR based classification algorithm works as the engine for the tool, and users can either use a default pretrained model or train their own model for crash classification. The tool presents the analysis results both in tabular form and on an interactive crash map for safety analysis in spatial context. Users can also download the full classification results for further analysis. The functions and analysis offered by CIEACT will provide safety practitioners and professionals quick access to the maximum amount of information stored in the texts of crash narratives and substantially reduce crash report review time.

17. Key Words Crash Data, Non-Motorist Safety, Pedestrian/Bicycle-Vehicle Crash, MV4000, DT4000, Data Quality, Crash Injury Severity Analysis	**18. Distribution Statement** No restrictions. This document is available through the National Technical Information Service. 5285 Port Royal Road Springfield, VA 22161

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages**	**22. Price**

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

DISCLAIMER

This research was funded by the Wisconsin Department of Transportation through the National Highway Traffic Safety Administration (NHTSA) under project FG-2020-UW-MILWA-05069 & FG-2021-UW-MILWA-05641. The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official views of the Wisconsin Department of Transportation (WisDOT) or National Highway Traffic Safety Administration (NHTSA) at the time of publication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BACKGROUND AND OBJECTIVE

The crash report is the primary source of data for analyzing a crash and identifying contributing factors. In Wisconsin, the WisTransPortal system maintains the database for all reported crash data. When a crash is reported, the law enforcement agency records crash information in appropriate fields of the structured data within the crash report. Each report also contains a narrative describing the crash. The narrative describes the sequence of events for all units involved in a crash and records additional information regarding citations, witnesses, drugs/medication, hazardous materials spillage from trucks and buses, trailer and towed, school bus information, etc. Information captured in the crash narrative is crucial, as it captures the unique and varying circumstances of each crash scene. The information is particularly helpful when analyzing misclassified or overlooked crashes.

Safety professionals mainly rely on the manual work of sifting through each narrative to extract relevant crash information. The manual review process is time-consuming and labor intensive. Additionally, the quality of a manual review is inconsistent, as it is subject to the reviewer's experience and judgement. In recent years, text mining and machine learning techniques have proven to be an efficient and effective method for automatically extracting crucial information from crash narratives. However, there currently is no tool available that would allow safety practitioners to utilize these techniques for crash narrative analysis.

The goal of this project is to develop an online Crash Information Extraction, Analysis and Classification Tool (CIEACT) using the Noisy-OR algorithm to develop the crash classification model. A user can either use the default pretrained model or train his/her own model for crash classification. The tool presents results both in tabular form and on an interactive crash map for safety analysis in spatial context. It also provides an opportunity to download the intermediate results of the user trained model. CIEACT provides safety practitioners and professionals with quick access to the maximum information stored in the texts of crash narratives. Furthermore, compared to traditional manual approach, this tool can substantially enhance crash report review quality and efficiency.

This report contains a detailed description of the features and functionalities of the CIEACT tool along with future scalability and recommendations. The following section describes the functionalities, framework, and system architecture of the tool. Section 3 explains the back end of the tool, such as the classification algorithm, database, and web framework. Section 4 details the front end of CIEACT, and section 5 includes the testing and evaluation of the tool. Section 6 reports a case study on secondary crash classification for the user trained model. Finally, conclusions and recommendations are given in section 7.

2. FUNCTIONALITIES, FRAMEWORK AND SYSTEM ARCHITECTURE

First, all functionalities of CIEACT were identified. Next, based on the functionalities, the framework and system architecture of the tool were designed. The tool was developed to be scalable so that more functionalities can be augmented in future. This section discusses the functionalities of the tool, explains CIEACT framework, and provides an overview of the system architecture of the tool.

2.1 Functionalities

The user survey results from "WisDOT DT4000 Crash Report Narrative Survey", conducted in the project titled "Using Text Data from the DT4000 to Enhance Crash Analysis", were used to determine the functionalities of CIEACT. Relevant web-based applications were considered before the functionalities of the tool were finalized. The following three points explain the core functionalities of CIEACT.

1) *Managing Uploaded Crash Data:* The tool provides users with the capability to upload crash data for analysis in the form of a csv file. Once uploaded the tool stores crash data in its memory or in a relational database (e.g., PostgreSQL) integrated with the tool. At present, some functionalities are performed using the memory of the tool while others use the database memory. In the future, it is possible to scale up the functionalities of the tool by managing all data in the database. This will also help to improve the tool's efficiency.
2) *Providing Models and Algorithms for Crash Classification:* CIEACT provides two options: (1) existing pretrained default models or (2) real-time trained models. To use the default model, users need only upload text data (e.g., crash narrative) for analysis and select among the available default models. On the other hand, to train a model on the go, users first need to upload training data and select the appropriate classification algorithm. The tool redirects the users to a model summary page once the model training is complete. The summary provides information about unigrams, bigrams and evaluation results based on different cut-off values and helps users decide whether to move forward with the trained model or retrain their model with a new or modified dataset. The users can select an appropriate cut-off value and use their trained models to classify their test data.
3) *Summarizing, Visualizing, and Downloading results:* The tool provides summary statistics of the user-trained model and presents the classification/analysis results in tabular form. Users can also visualize the locations of all crashes on an interactive crash map for spatial analysis. Finally, CIEACT provides users with the option to download the results for further review and analysis.

2.2 Framework

The framework of the tool was designed after its functionalities were identified. The framework was designed by preparing an information architecture, creating a wireframe, and then determining the technical system requirements. The following subsections discuss the three aspects of framework design.

2.2.1 Information Architecture

Information architecture (IA) is the combination of organizing, labeling, searching, and navigating systems within the webtool. The IA for this project was designed to provide a comfortable user experience. Therefore, all necessary information was considered during the IA design, including textual information (such as text in header, footer, headline and body, information in the dropdown menu, links to other page, etc.), logos, user guides, project descriptions, any external links, etc.

2.2.2 Wireframe

A visual representation of the different web pages of the tool was created by focusing on functionalities, content, layout, and relationships among the different pages. The wireframe helped the project team get feedback at an early stage of the project and thus guided the programmers during the tool's development.

2.2.3 Technical System Requirements

Technologies such as the programming language, software, and database system were chosen based on the information architecture and wireframe. Django, a free and open-source Python-based web framework, was used to develop CIEACT. Django is fast to implement and exceedingly scalable. "PostgreSQL[1]" was selected as the database management system. PostgreSQL is an open-source system that is reliable, scalable, and secure (Douglas & Douglas, 2003).

2.3 System Architecture

Below is a brief overview of the tool's system architecture (Figure 1).

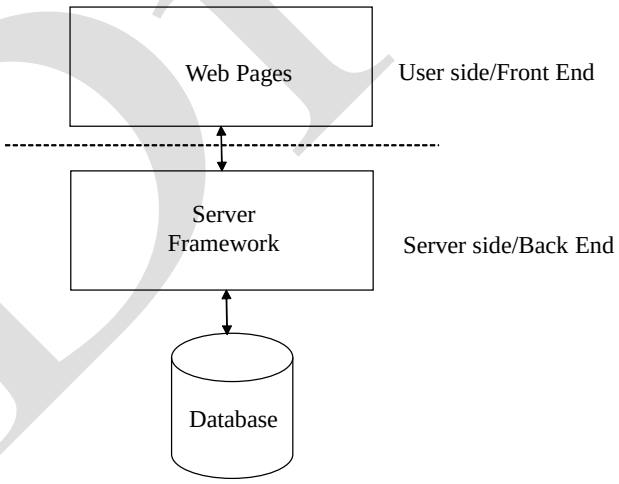


Figure 1: System Architecture of CIEACT

Figure 1 shows that the tool has two main parts which are divided by the dashed black line. The user side, or the "front end", is the part that is accessible by users on an electronic device such as a desktop computer, cell phone, laptop, etc. The front end links to different web pages through

[1] https://www.postgresql.org/

which a user can navigate the tool. Users can classify/analyze their text data, conduct a spatial analysis on map, and download analysis results using the functionalities in the front end of the tool.

The “back end” is the lifeline of the tool that consists of the server framework and database. The back end manages data flow and storage and ensures proper functioning of all of the different algorithms for model training and testing. Additionally, the back end communicates with the front end to send and receive information and display results on the web pages. The next two sections explain the front end and back end of CIEACT in detail.

3. BACK-END: CLASSIFICATION ALGORITHM, DATABASE AND WEB FRAMEWORK

The efficiency and user experience of the tool depend on the proper functioning of the back end. This section describes the classification algorithm, database management system, and web framework of the CIEACT.

3.1 Classification Algorithm

A Noisy-OR based classification algorithm was implemented to classify crash narratives. The Noisy-OR method calculates the probabilities of specific types of crash narratives by combining the probability scores of unigrams (words) and bigrams (two consecutive words) in the narrative. Noisy-OR is basically a probabilistic extension of logical "or" (Oniśko et al., 2001; Vomlel, 2006). If an input has a high probability score (such as a value close to1), then the combined probability in Noisy-OR becomes high. The combined probability in Noisy-OR is higher when more input probabilities are high.

To apply the Noisy-OR classifier to crash narratives, the algorithm computes probabilities of unigrams and bigrams and then combines these probabilities using the Noisy-OR method to calculate the final classification score. The method is explained in detail below.

For every unigram and bigram *w* in the corpus, the method first computes the probability that if it is present in a narrative, then the narrative is positive, i.e., P (positive | *w*). Then, the method computes the probability using simple frequency counts. Equation 1 shows the calculation.

$$Probability\ Score\ (w) = \frac{\text{Positive Count}(w)+1}{\text{Positive Count (w)}+\text{Negative Count(w)}+2} \quad (1)$$

where *w* is a unigram, or a bigram. *Positive Count* means the number of occurrences of *w* in the positive narratives. Similarly, the *Negative Count* indicates the number of occurrences of *w* in the negative narratives.

The probability score of the word *w*, which is the ratio of the frequency of the word *w* in positive narratives to the frequency of the word *w* in both positive and negative narratives. Next, a smoothing is applied by adding one in the numerator and two in the denominator of the Equation. This simple version of Laplace smoothing assumes *w* occurred at least once in a positive narrative and in a negative narrative. Smoothing done in this way ensures that among the unigrams, and bigrams with zero negative counts, those with higher positive counts receive higher probability scores. Otherwise, they will all will receive an unrealistic probability score of 1 because they occurred in a few positive narratives and no negative narratives.

Additionally, minimum positive count can be fixed to control some words that provide a high probability but are very unlikely to occur in positive cases (Sayed et al., 2021). In the case of words that appear in both positive and negative narratives with a very high frequency (Count), it is likely to reduce the probability of that specific word. For example, let a unigram '*unit*' appears in the narratives of a specific type of crash (positive case) 110,933 times and in all the other narratives (negative cases) that exclude that specific crash 1,000,904 times. Then according to Equation 1, the probability score for that unigram is 0.099. This indicates that the word is not relevant for the

classification task. On the other hand, if a unigram/ bigram appears frequently in both positive and negative narratives but has a higher frequency in positive narratives, Equation 1 gives a good probability score to the corresponding unigram/ bigram. For example, if the unigram '*inattentive*' appears in the narratives of a specific crash 2743 times and in all the other narratives 1808 times, Equation 1 yields a probability of 0.6023, indicating that the word is relevant for the classification task.

To classify a given narrative as positive or negative, this method computes the probability of a positive narrative by combining the probability scores of the unigrams and bigrams present in that narrative. The method needs to compute P (positive|w_1,w_2,...,w_n), where w_1..w_n are unigrams and bigrams present in the narrative. It computes the probability by combining the probabilities P(positive|w_1), P(positive|w_2),…, P(positive|w_n), which have been computed as described earlier.

Noisy-OR is a method of combining probabilities (Zagorecki & Druzdzel, 2004) which is commonly used in Bayesian networks (Oniśko et al., 2001; Vomlel, 2006). Instead of true/false values in Noisy-OR, the inputs and output are probabilities (hence termed "noisy"). Analogous to logical "or", in Noisy-OR, if any one of the input probabilities is high (i.e., close to 1), then the combined probability is high. But unlike logical "or", the combined probability is even higher when more input probabilities are high. The combined probability is low (i.e., close to 0) only when all of the input probabilities are low. The Noisy-OR combined probability is mathematically computed as shown in Equation 2, where the probability score of a narrative is calculated by combining the probability scores of unigrams or bigrams occurring in it.

$$Noisy - OR\ Proability\ Score\ (N) = 1 - \prod_{i,j=1}^{n}(1 - P_i)^j \quad (2)$$

where N is a given narrative, P_i indicates the probability score of i^{th} unigrams or bigram as computed from the training data using Equation 1, and j means the number of occurrences of that i^{th} unigram, or bigram in the crash narrative N.

It should be clear from Equation 2 that if there is no unigram or bigram in a narrative with a high probability score, the probability score of the narrative will be close to zero. On the other hand, a single unigram or bigram with a high probability score will result in a high probability score for the entire narrative. Furthermore, more unigrams and bigrams with high probability scores make the combined probability score higher.

3.2 Database

CIEACT was developed by integrating the PostgreSQL database that complies with SQL (Structured Query Language). All necessary intermediate data from the default model was stored in this database. As an example, Figure 2 shows the necessary data fields, including the order and format of the fields for the default model stored in the PostgreSQL database. The numeric data stored in the database was fixed to 6 decimal values. The existing default model can be updated by replacing existing word probability data with updated word probability data in the database. In future, it is possible to upload the word probability data for new crash types to the database using a similar script as shown in Figure 3 below.

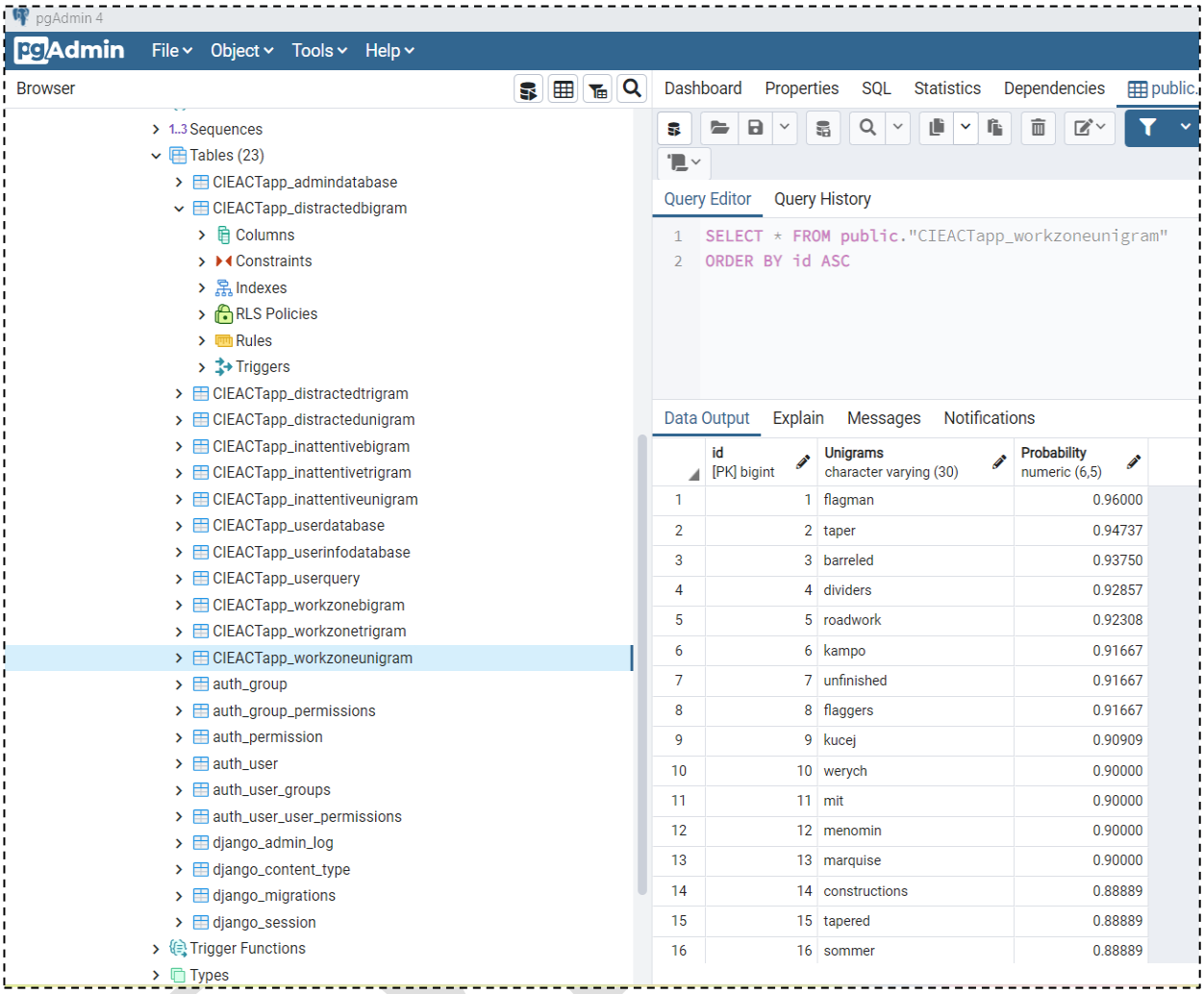


Figure 2: Unigrams and Corresponding Word Probability Scores

```
copy  "CIEACTapp_workzoneunigram" from '<directory>\unigram_probability_0.25_WZ.csv'DELIMITER ',' CSV header;
copy  "CIEACTapp_workzonebigram" from '<directory>\bigram_probability_0.25_WZ.csv'DELIMITER ',' CSV header;
copy  "CIEACTapp_workzonetrigram" from '<directory>\trigram_probability_0.25_WZ.csv'DELIMITER ',' CSV header;
```

Figure 3:SQL Script to Upload the Word Probability csv Data File (to be used with the Default Model)

3.3 Web Framework

CIEACT was developed with the Python based web framework Django[2]. This section gives a detailed discussion of the framework, which is a combination of three basic parts: communication, data storage and management, and presentation.

Communication: In this step, the user communicates with the server through HTTP (Hypertext Transfer Protocol). The protocol has two fundamental parts: requests (client to server) and responses (server to client). A request from the client/user to the server is done using a URL (Uniform Resource Locator) which is the 'path' to the document, service or function being requested. The response from the server side is an HTML (Hypertext Markup Language) document such as a web page. The request and response are represented as Python objects with attributes for the varying pieces of data and methods for more complex operations.

Data storage and management: The CIEACT tool allows the user to upload data in csv format. It utilizes memory to store information from the user trained model and uses a relational database to store data from the default model through Django. Django provides a powerful ORM (Object-Relational Mapper) for storing and managing these data. The ORM represents the database as a code object where Python classes represent tables, objects represent individual rows within those tables, and the columns of the table are attributes of those objects. Also, the user can download the data produced in the form of classification and analysis results in a csv file.

Presentation: The final piece of the web framework is presenting the information requested and/or returned via HTTP and queried from the database. This is done using templates written in HTML (Hypertext Markup Language), along with other languages like JavaScript for dynamic browser-side functionality and CSS (Cascading Style Sheets) for visual styling.

Before going into further details of the framework, we will discuss how the Django-based web framework follows the Model-View-Controller (MVC) paradigm. In this paradigm, the web framework is segregated into the model, view, and controller. The model controls the data, whereas the view defines how to display the data. Finally, controller mediates between the two and enables the user to request and manipulate the data.

Figure 4 shows the interactions among different layers in an MVC pattern. Django adheres to this MVC pattern, but it does so in a slightly different manner than usual. Models in Django deal solely with data passing into and out of a database. The models work in a fashion similar to other web frameworks, but views in Django work like the controller aspects of MVC. The views are Python functions, which tie together the model layer and the presentation layer (which consists of HTML and Django's template language). In other words, Django splits the presentation layer in twain with a view defining what data to display from the model and a template defining the final representation of that information. As for the controller, the Django framework itself serves as a type of controller by providing mechanisms to determine what view and template are used to respond to a given request.

[2] https://www.djangoproject.com/

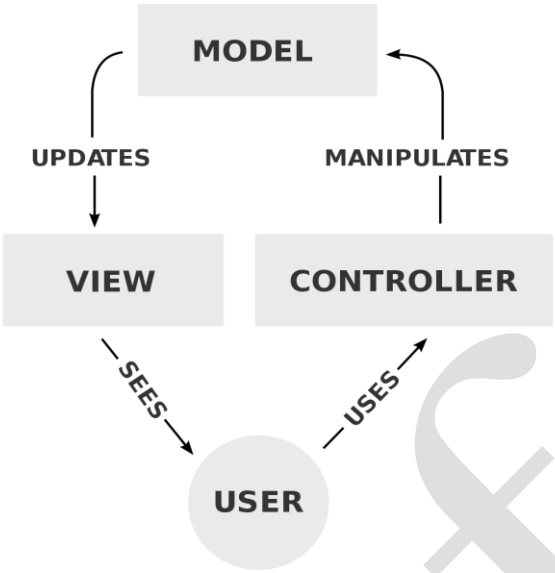


Figure 4: Diagram of Interactions within an MVC Pattern for the Django-based Web Framework

As explained above, Models and Views were developed for the webtool. The built-in controller capability of the Django-based web framework was used to complete the full MVC paradigm. In addition, different templates were designed to be used by the functions defined inside Views. These are briefly discussed below.

Models: The model is the foundation layer of the Django. Views and templates can change according to how data enters and leaves the model or how data is presented, but the model is relatively set in stone. The model layer makes heavy use of ORM (Object-Relational Mapper). We developed the models for the webtool according to its functionalities.

Views: Views form much (sometimes most or all) of the logic behind the functions of the webtool. There are some Python functions linked to one or more defined URLs, which return HTTP response objects. The authors did the coding to develop all required views for the webtool.

Templates: Templates are HTML text documents with special formatting denoting where to output dynamic values, enabling simple logic constructs such as loops. Different templates were developed based on the functionalities of the webtool. The functions in views call specific templates to display information via an HTML document.

3.3.1 Overview of the Framework

Figure 5 shows the overall framework architecture of the webtool. The black dashed rectangle represents the Django-based web framework of CIEACT. The figure shows that once the user generates the HTTP request via the web server, they go to the Request middleware layer. It is then dispatched based on URLconf patterns to the appropriate View. In the next step, Views perform

the core part of the work which requires the use of models and/or templates to create a response. Lastly, the response goes through one more layer of middleware that performs any final processing before returning the HTTP response back to the web server. The response is then forwarded to the user via the web server.

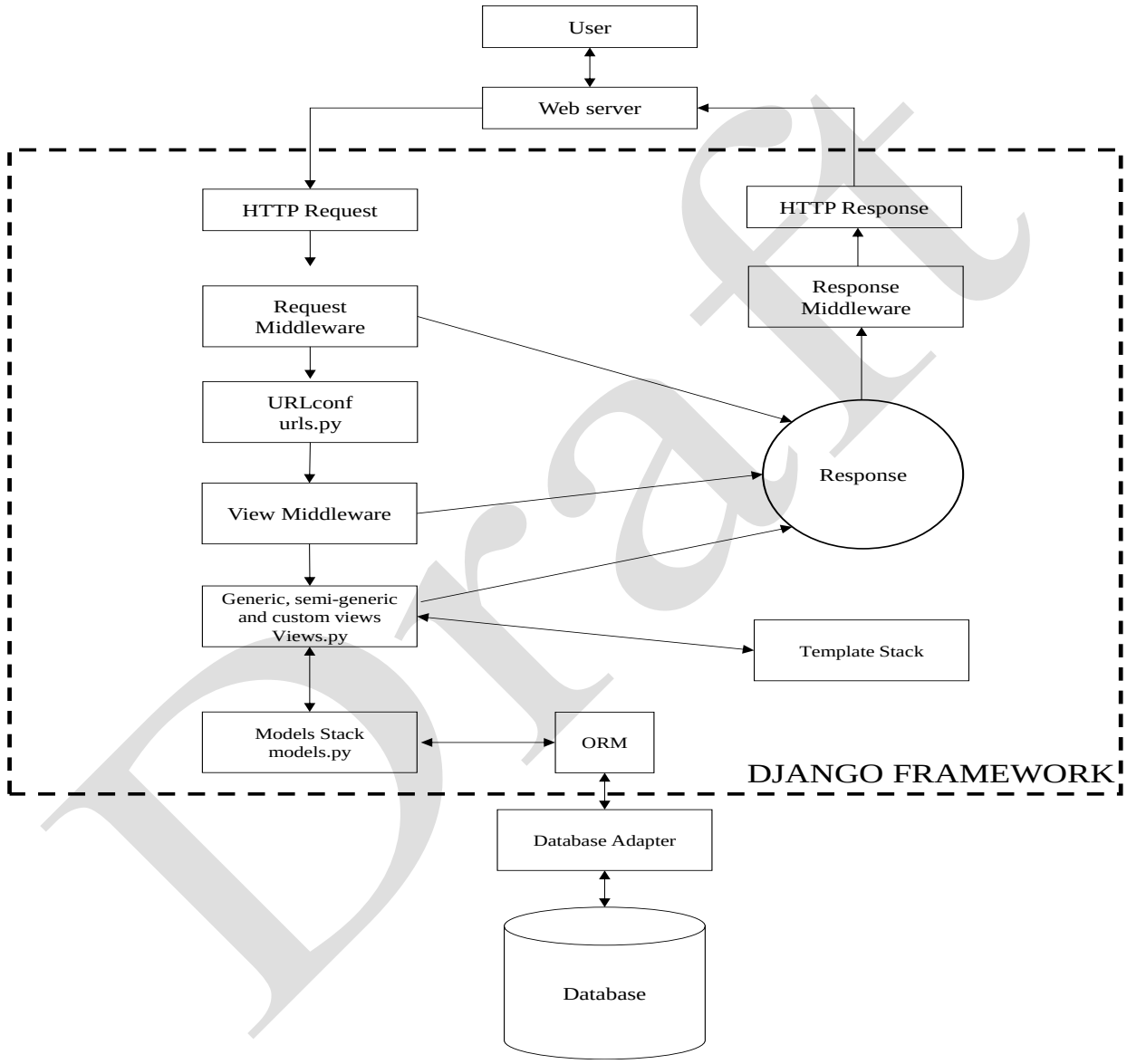


Figure 5: Overview of the Web Framework for CIEACT

3.3.2 Logical Diagram of CIEACT

Figure 6 below shows the logical diagram of CIEACT interacting with different Views, Algorithms, and Templates. Their functionalities are discussed in Table 1.

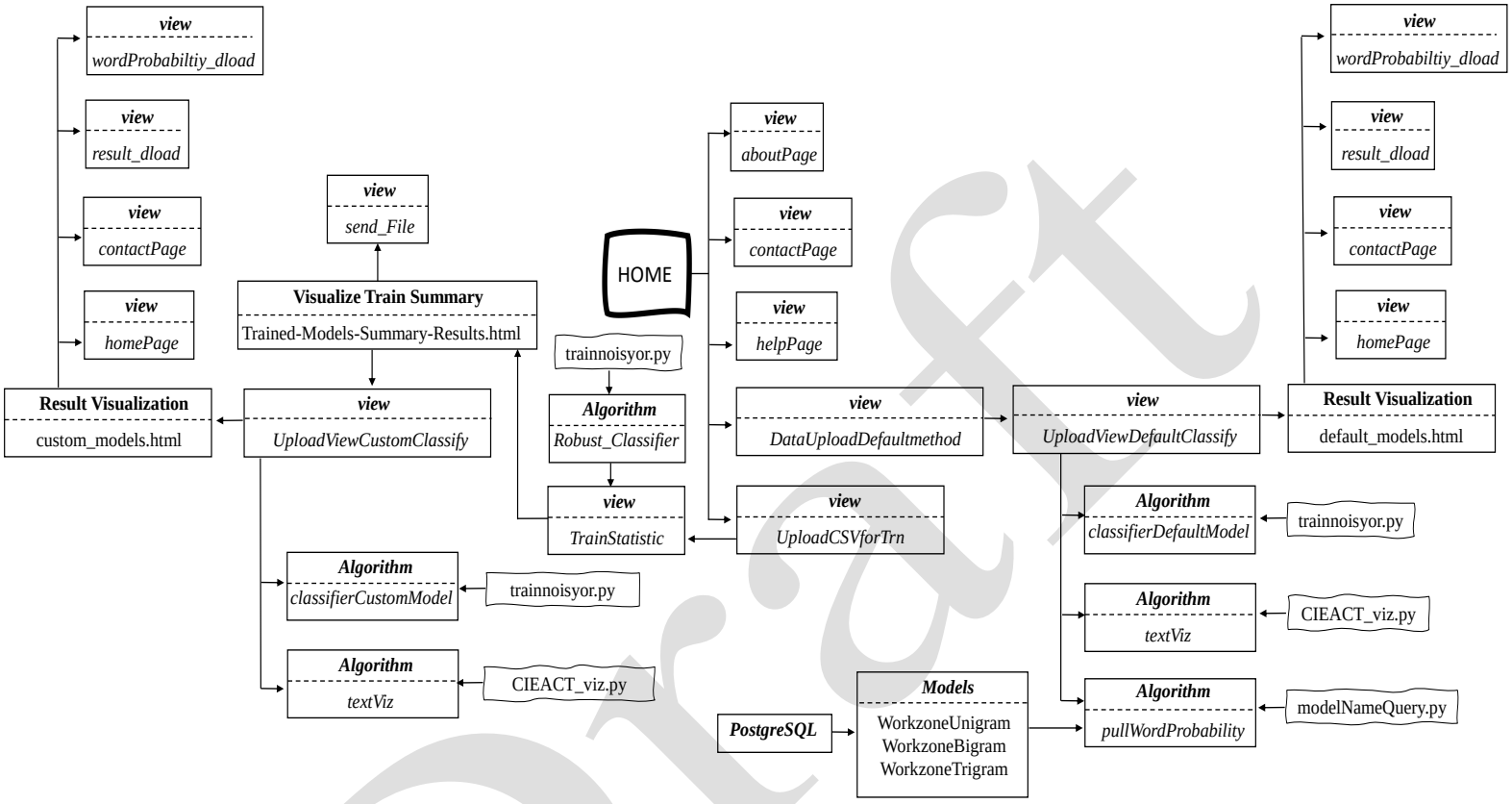


Figure 6: Logical Diagram of Functionalities of CIEACT

Table 1: Descriptions of the Views and Algorithm in the Logical Diagram

Views	Description
aboutPage	It provides an overview of the tool.
contactPage	It directs users to the project team for further help.
helpPage	It provides detailed information on how to use the tool.
DataUploadDefaultmethod	It navigates users to the default model.
UploadCSVforTrn	It checks and accepts the user-uploaded csv data for training a new model. If the upload is successful, the classification and evaluation metrics algorithms are executed, along with the functionality to download the word probability and proceed to the subsequent step for text classification.

Views	**Description**
UploadViewDefaultClassify	It accepts the csv data uploaded by the user for the default model. It then executes the classification and visualization algorithms. In addition, it provides the option to download the word probability results.
UploadViewCustomClassify	It accepts csv data uploaded by the user for their trained model. The classification and visualization algorithms are subsequently executed. In addition, it offers the ability to download classification results.
wordProbability_dload	It provides download functionality for word probability data.
result_dload	It provides download functionality for classification results.
Algorithms	**Description**
classifierDefaultModel	It imports and runs the algorithm of the default model from *trainnoisyor to classify text narratives.*
textViz	It provides functionalities for visualizing results in tabular format imported from *CIEACT_viz*.
pullWordProbability	It provides functionality for acquiring the word probability of the default model from the database imported from *ModelNameQueary*.
Robust_Classifier	It provides classification, evaluation matrices and word probability algorithms imported from *trainnoisyor*.
classifierCustomModel	It provides the functionality of a user-trained custom classifier imported from trainnoisyor.

4. FRONT-END: WEB PAGES OR USER INTERFACE

The front end of CIEACT is the interface displayed to users on an electronic device while operating the tool. Web pages were created for navigation within the tool. The web pages were created with the intention to make them interactive and informative for the user. This section describes the organization, functions, and relation of the tool's various web pages.

4.1 Homepage

The basic functionalities of the homepage provide introductory information about the tool and directions for users to help them navigate the web pages. The home page includes a navigation bar at the top, links to various pages, and a basic description of the tool. The navigation panel at the top of the page links the user to 'About', 'Contact', and 'Help' pages. It also provides a selection option and links to the default model page and the train model page. Figure 7 below is a screenshot of the homepage.

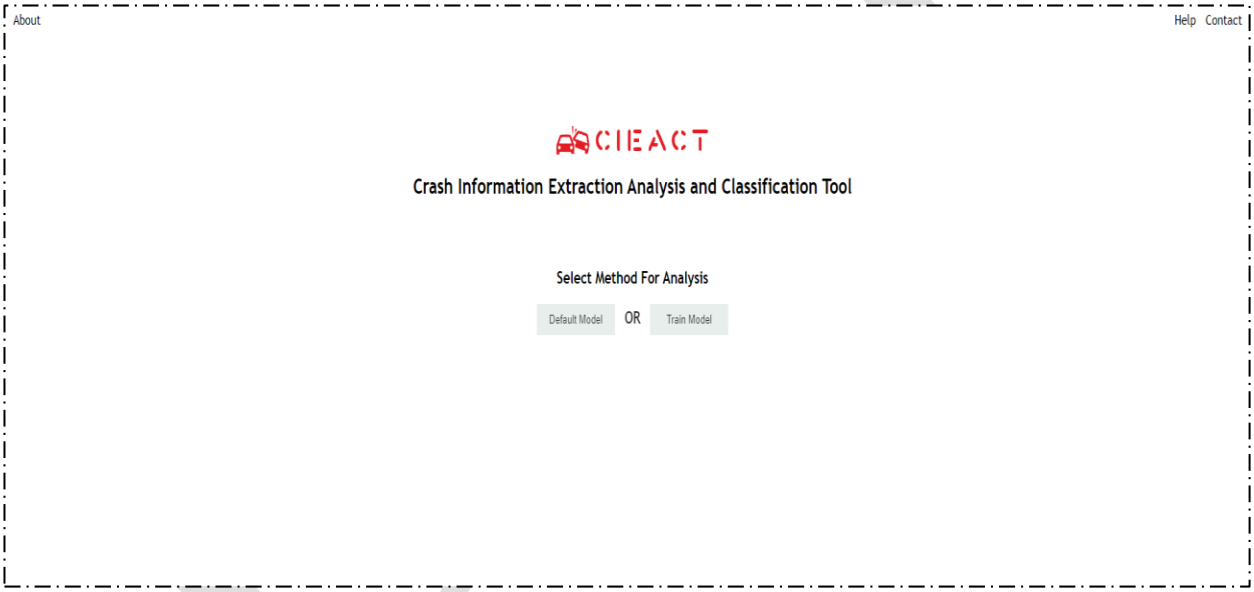


Figure 7: Screenshot Showing the Homepage of CIEACT

As mentioned before, from the home page a user can navigate to other web pages through links provided in the top navigation bar. The 'About' link redirects users to a page with a description of the tool. Figure 8 below shows the screenshot of the 'About' page.

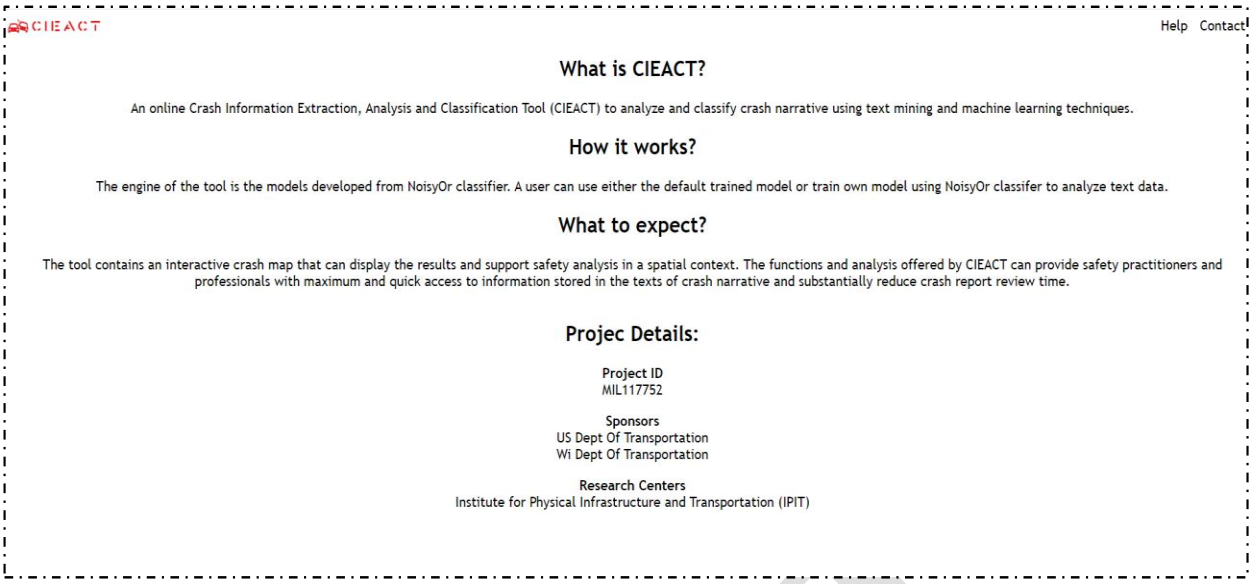


Figure 8: Screenshot Showing the About Page of CIEACT

The 'Contact' page provides the necessary contact details for the project team. Users may share concerns, queries or suggestions regarding CIEACT using the provided contact information. Figure 9 is a screenshot of the 'Contact' page.

Figure 9: Screenshot Showing the Contact Page of CIEACT

The ‘Help’ link provides detailed information on how to a) use the tool, b) navigate through the pages, c) interpret training and testing results, d) visualize results on the map, and e) download results for future analysis. It is recommended for a first-time user of the tool to review the ‘Help’ page before using the tool. Figure 10 below shows a screenshot of the ‘Help’ page, which opens in a new window that allows users to review documentation while using the tool.

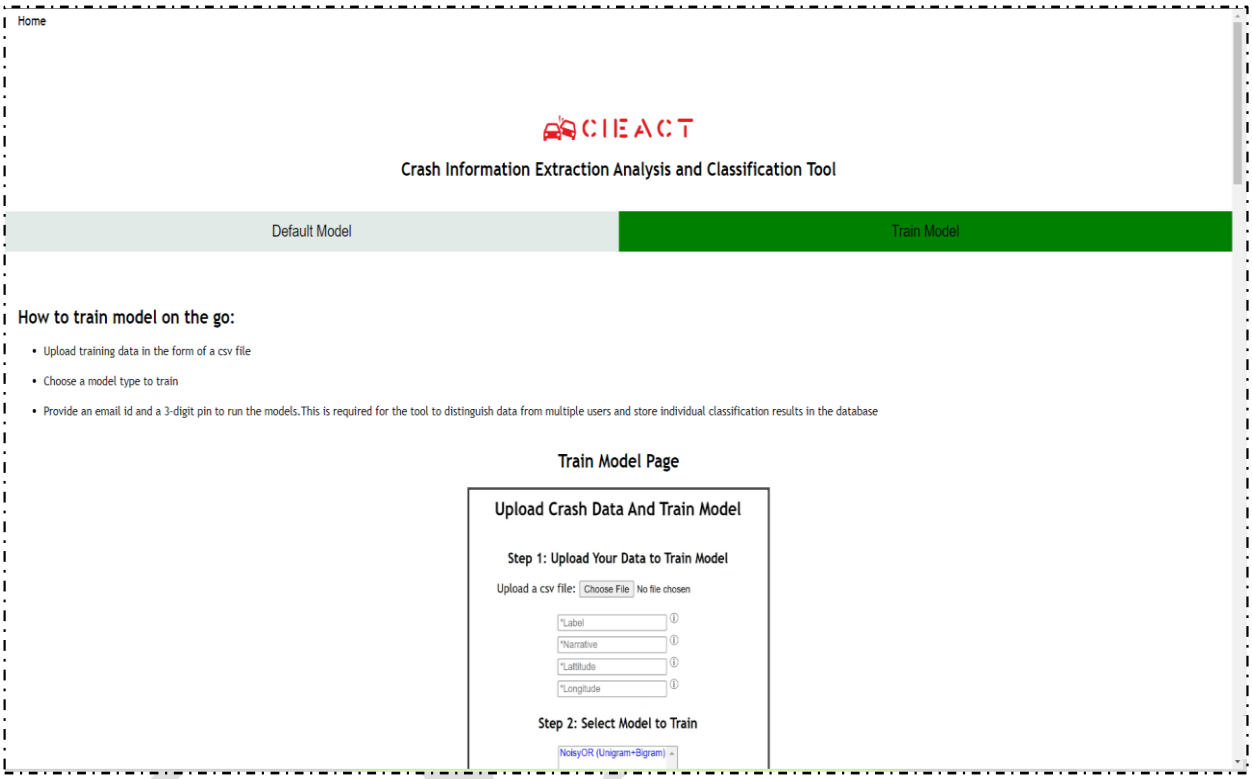


Figure 10: Screenshot Showing the Help Page of CIEACT

4.2 Default Model Page

The ‘Default Model’ link on the homepage redirects users to the Default Model page. This page provides users with options to upload crash data or narratives in the form of a csv file for analysis in CIEACT. Users can then choose from default classification algorithms (such as, Noisy-OR for work zone classification) and run models for crash classification. Each user needs to provide an email ID and a 3-digit pin in order to run the models. The ID and pin are required so the tool can distinguish data from multiple users and store individual classification results in the database. Once the model run is complete, the tool redirects users to a result visualization page which displays the results. The contents and functionalities of the visualization page are described later in this section. Figure 11 below shows a screenshot of the ‘Default Model’ page.

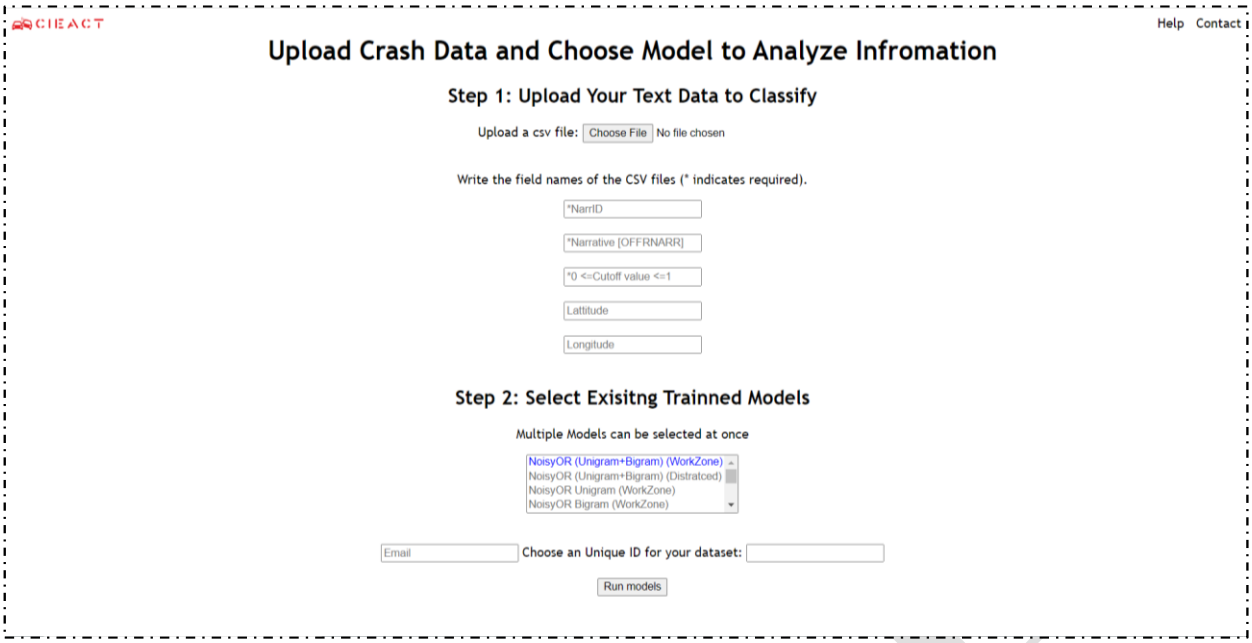


Figure 11: Screenshot Showing the Default Model Page of CIEACT

4.3 Train Model Page

The 'Train Model' link on the homepage navigates users to the Train Model page where users can train a crash classification model in real-time by using their own training data. The user can use the training summary to choose a cut-off value for crash classification. The 'Train Models' button on this page starts the model training. Once the training is complete, the tool redirects users to a results summary page. Figure 12 shows the screenshot of the 'Train Model' page.

4.4 Result Summary Page

The Result Summary page shows a table with summary statistics of the crash narrative. It also presents the evaluation metrics of training for several cutoff values. The table with summary statistics of the crash narrative provides the top 10 unigrams and bigrams with their probability scores. This information helps the user understand the trained model. It is expected that the top unigrams and bigrams will be relevant to the target class. The evaluation metrics can help the user to choose an appropriate cutoff value to classify their test data. The cutoff value is the minimum probability score of a word that must be used in the model (unigrams and bigrams). The trained model will perform well if the top unigrams and bigrams are relevant to the target class, and a proper cutoff value is used for the test data. This page also provides an option to download words probability results for further analysis. Figure 13 below shows a screenshot of the 'Result Summary' page.

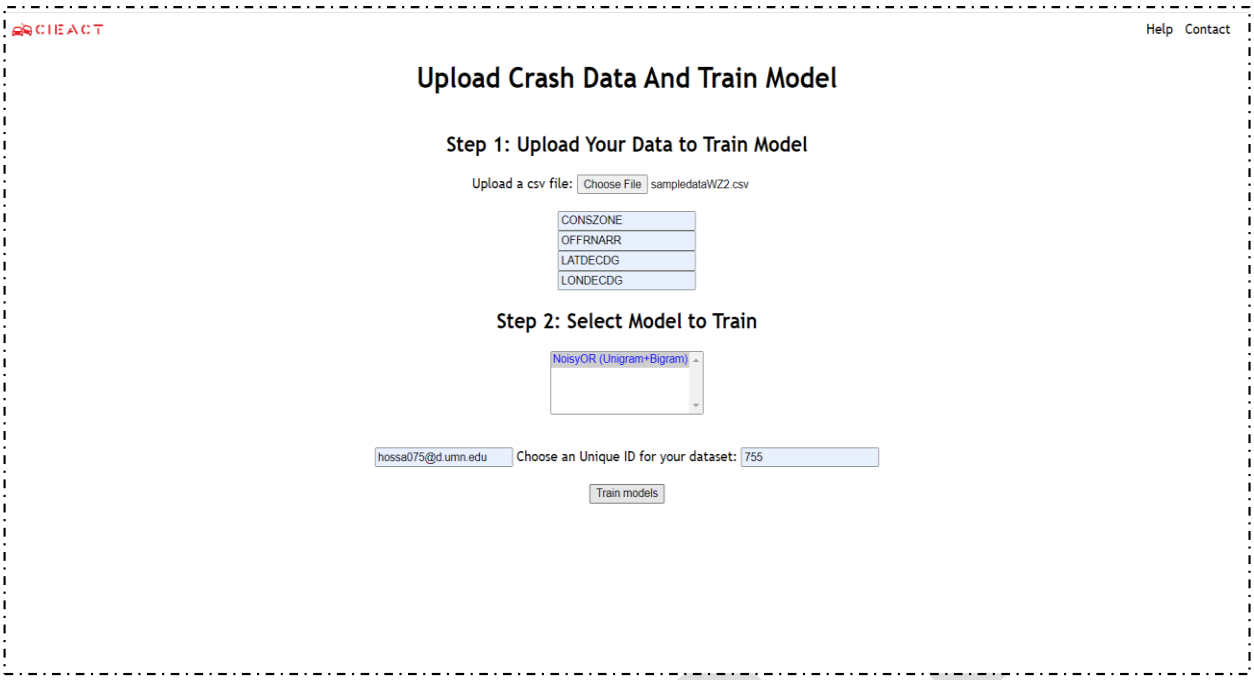
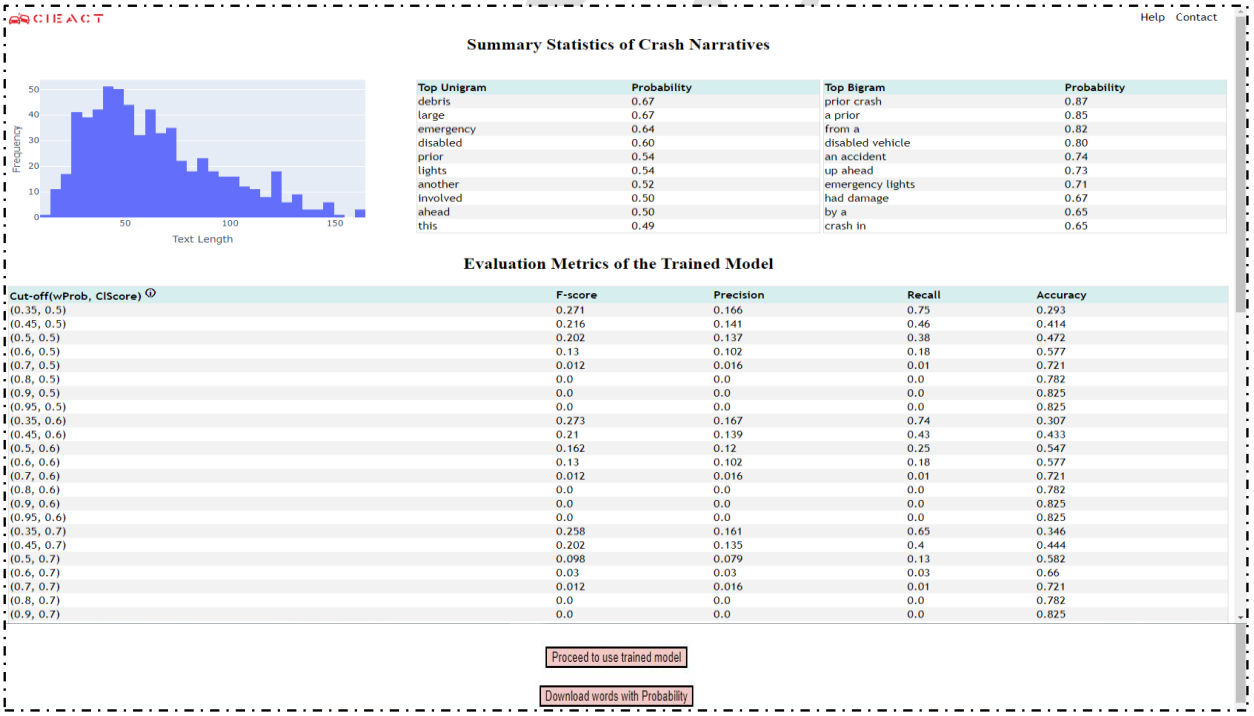


Figure 12: Screenshot Showing the Train Model Page of CIEACT



Top Unigram	Probability	Top Bigram	Probability
debris	0.67	prior crash	0.87
large	0.67	a prior	0.85
emergency	0.64	from a	0.82
disabled	0.60	disabled vehicle	0.80
prior	0.54	an accident	0.74
lights	0.54	up ahead	0.73
another	0.52	emergency lights	0.71
involved	0.50	had damage	0.67
ahead	0.50	by a	0.65
this	0.49	crash in	0.65

Cut-off(wProb, CIScore)	F-score	Precision	Recall	Accuracy
(0.35, 0.5)	0.271	0.166	0.75	0.293
(0.45, 0.5)	0.216	0.141	0.46	0.414
(0.5, 0.5)	0.202	0.137	0.38	0.472
(0.6, 0.5)	0.13	0.102	0.18	0.577
(0.7, 0.5)	0.012	0.016	0.01	0.721
(0.8, 0.5)	0.0	0.0	0.0	0.782
(0.9, 0.5)	0.0	0.0	0.0	0.825
(0.95, 0.5)	0.0	0.0	0.0	0.825
(0.35, 0.6)	0.273	0.167	0.74	0.307
(0.45, 0.6)	0.21	0.139	0.43	0.433
(0.5, 0.6)	0.162	0.12	0.25	0.547
(0.6, 0.6)	0.13	0.102	0.18	0.577
(0.7, 0.6)	0.012	0.016	0.01	0.721
(0.8, 0.6)	0.0	0.0	0.0	0.782
(0.9, 0.6)	0.0	0.0	0.0	0.825
(0.95, 0.6)	0.0	0.0	0.0	0.825
(0.35, 0.7)	0.258	0.161	0.65	0.346
(0.45, 0.7)	0.202	0.135	0.4	0.444
(0.5, 0.7)	0.098	0.079	0.13	0.582
(0.6, 0.7)	0.03	0.03	0.03	0.66
(0.7, 0.7)	0.012	0.016	0.01	0.721
(0.8, 0.7)	0.0	0.0	0.0	0.782
(0.9, 0.7)	0.0	0.0	0.0	0.825

Figure 13: Screenshot Showing the Result Summary Page of CIEACT

The Result Summary page provides users with a link that redirects them to a page where they can upload test data in the form of a csv file. The user can select a cutoff value and run the trained model to classify their data. Figure 14 shows the screenshot of this page.

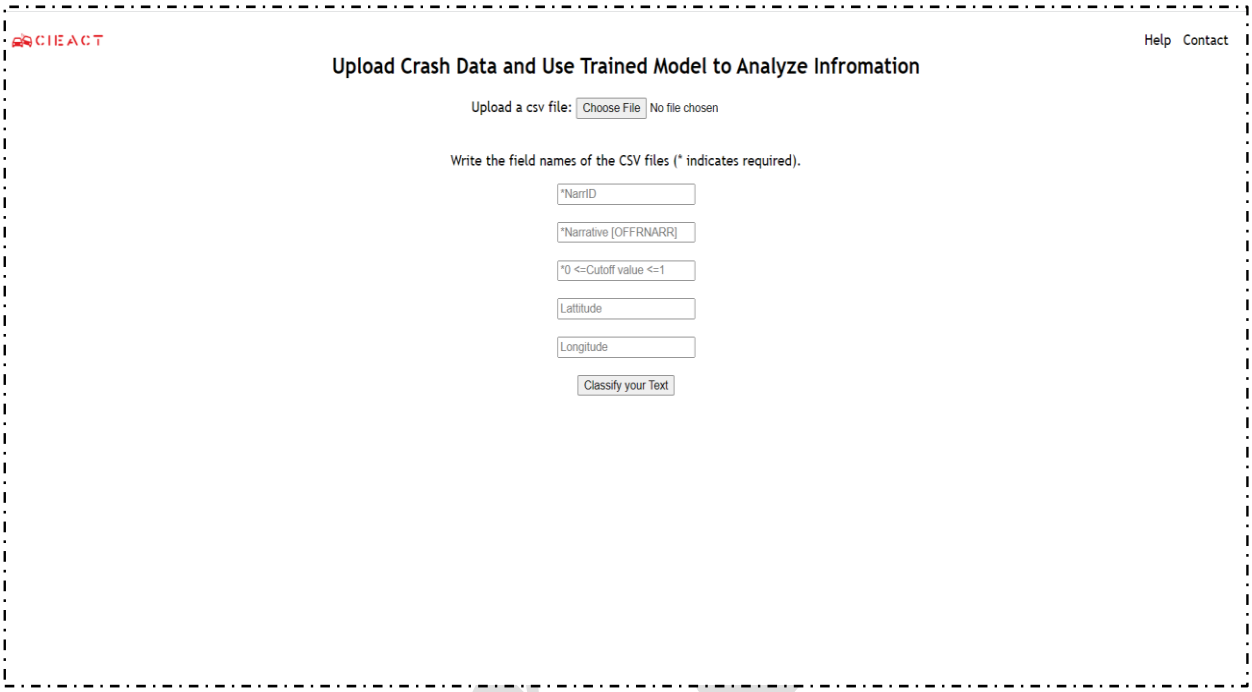


Figure 14: Screenshot Showing the Page that Classifies Crash Data in CIEACT when Working with the User Trained Model

4.5 Result Visualization Page

The Result Visualization page displays classification results from the default model and the trained model. The tool automatically redirects users to this page after it completes running a model. On this page, CIEACT shows the detailed results of the crash analysis. Also, the tool presents the significant unigrams and bigrams from each crash narrative using color coding. A user can investigate the crash narratives quickly by viewing the tabular results. This page also provides the user with the option to see the location of any crash on a map. The map is interactive, and users can zoom in to get an idea of the surroundings of the crash location. There is also an option to visualize all crash locations on the map. These visualizations help users see the spatial distribution of crashes or find a pattern of crash locations. The navigation bar at the top right of the page provides a download option so users can download the full classification results. Figure 15 shows a screenshot of the 'Result Visualization' page.

Figure 15: Screenshot Showing the Result Visualization Page of CIEACT

5. EVALUATION AND TESTING

In the evaluation and testing phase, results from the default and custom trained models were evaluated. In addition, the tool for browser compatibility – HTML and CSS syntax validity – was also tested. The following two subsections provide details on the evaluation and testing conducted on CIEACT.

5.1 Evaluation of the Output of the Tool

While doing the original offline code for the Noisy-OR algorithm, the code was not properly optimized and was divided into multiple parts. In CIEACT, we converted all the intermediate steps into several functions to make the program run faster and the code easier to reuse. To ensure proper implementation of the crash classification model in CIEACT, we validated the classification results by comparing them with the offline model.

For the user trained model, a two-step validation process was followed. First, a Noisy-OR model was trained both in CIEACT and offline (outside the tool) using a sample dataset with 10,000 crash narratives and other structured data fields. Zero differences in the word probability scores were observed in either case, indicating that model training within CIEACT was successful. Next, classification scores for the trained model in CIEACT were compared with scores for the offline model. No difference was observed, thus confirming that the Noisy-OR implementation in the CIEACT was successful.

In case of the default model, a single-step validation process which consisted of cross-checking the final classification results was used. This is because the tool took word probabilities stored in the PostgreSQL database of CIEACT from a pretrained offline model. Results in CIEACT matched the results from the offline model, confirming the correct implementation of the default method in CIEACT.

5.2 Browser Compatibility, HTML and CSS Syntax Validity Testing

The tool was tested for browser compatibility using the following platforms/versions: *Internet Explorer (versions 7.x and 8.x); Firefox (version 3.x); Safari (version 4.x); Opera (version 10.x); Chrome (version 5.x).* The testing devices include *Desktop computer; Smart phone; Tablet.* We found that the tool had no issue running in different browsers as well as on different devices. The HTML and CSS syntax validity were tested on different platforms as well, and the tool was tested in the cloud using Heroku[3]. All tests showed the tool was working properly.

[3] https://www.heroku.com/home

6. CASE STUDY

A case study was used to test CIEACT's effectiveness in solving a new problem. The case study sought to identify secondary crashes within crash narratives. A secondary crash is defined as any crash beginning with the time of detection of the primary incident where the collision occurs, either within the incident scene or within the queue, including the opposite direction, resulting from the original incident (FHWA, 2020).

Data were collected from the WisTransPortal data hub of the Wisconsin Department of Transportation through the UW-Madison TOPS lab. Positive cases were selected from the training dataset using the data field "SECDFLAG" which flagged any secondary crash in the database. An example crash narrative that is flagged as a secondary crash is "*Unit 1 was traveling northbound on i-41 in lane 1 at apple creek rd. A metal beam from the median cable barrier was laying in lane 1 of i-41 northbound from a* ***previous crash*** *in the median. Unit 1 drove over the metal beam, striking the undercarriage of the vehicle. The beam became lodged under the engine of the vehicle, disabling the engine. Unit 1 moved to the right shoulder and came to rest facing north on i-41 northbound*." The narrative implied that an incidence related to a previous crash was behind the occurrence of the crash, and therefore it was classified as a secondary crash.

Two training datasets were considered for this case study: one with 570 crash narratives and 100 positive cases (flagged secondary crash), and the other with 5545 crash narratives and 1042 positive cases. All positive cases were taken from the secondary crashes that occurred in Wisconsin from 2018-20 in non-intersection areas of divided highways. The cases did not involve deer. The negative cases were chosen randomly from a pool of 61,960 non-intersection, non-deer, divided highway crashes from 2018-20 in Wisconsin. Trained models were tested using a test dataset containing 40,516 crash narratives that occurred between January 1, 2021, and June 16, 2021. The remainder of this section discusses the model training summary, training results with word probability of unigrams and bigrams, and testing results for the case study.

Case study with 570 training data, 100 positive cases

The model training was conducted on an ACER laptop that had Intel(R) Core (TM) i5-9300H CPU @ 2.40GHz, 2.40 GHz processor, 8.00GB of RAM and 64-bit Windows operating system. The model training was completed in approximately 3 seconds. Table 2 shows the top 20 unigrams and bigrams with their probability scores from the trained model. It shows that only nine unigrams and all bigrams have probability scores greater than 0.70. The majority of the words in the tables are irrelevant to secondary crashes. The top three bigrams pertaining to secondary crashes are "previous crash," "prior crash," and "a prior." Only the unigram "previous" at the top of the list could be related to our intended crash. This observation suggests that the probability values of the words greater than or equal to 0.846 will be an appropriate cutoff value for identifying secondary crashes from narratives. Due to the small training dataset, it is possible that important information regarding secondary crashes is missing; consequently, we should not draw a definitive conclusion from this dataset.

Table 2: Top 20 Unigrams and Bigrams by Probability Score from Model Training

Unigrams	Probability	Bigram	Probability
Previous	*0.867*	*previous crash*	*0.917*
International	*0.800*	*prior crash*	*0.867*
Initial	*0.750*	*a prior*	*0.846*
Ln	0.727	from a	0.824
Separate	0.714	following unit	0.800
Four	0.714	disabled vehicle	0.800
Trailers	0.714	the debris	0.800
Jones	0.700	traffic due	0.800
Crashes	0.700	reported that	0.800
Debris	0.667	crash he	0.778
Large	0.667	that there	0.778
Something	0.667	unit 01	0.778
Less	0.667	stopped traffic	0.778
Reduced	0.667	units came	0.778
Hazard	0.667	contact unit	0.778
Pressed	0.667	traffic crash	0.769
Six	0.667	1 semi	0.750
Temporary	0.667	accident in	0.750
Hauling	0.667	stop on	0.750
Gmc	0.667	the international	0.750

In addition to word probability, the tool provided summary statistics of the model training for different cut-off values of word probability and classification score, shown in Table 3. The emphasis was on recovering more secondary crashes in order to select cut-off values. Therefore, we were more lenient in accepting false positive (instead of false negative) cases. According to the table, the best results were found with cut-off values of 0.7 for word probability and 0.8 for classification score. Thus, a word probability of 0.7 was used as the cut-off value for test dataset. The tool took approximately 10 seconds to upload test data and run the model.

Table 3: Trained Model Statistics with Different Cut-Off Values for Word Probability (W-Prob) and Classification Score (Cl-Score)

Cut-Off (W-Prob, Cl-Score)	F-Score	Precision	Recall	Accuracy
(0.25, 0.5)	0.299	0.176	1	0.177
(0.35, 0.5)	0.306	0.181	1	0.205
(0.45, 0.5)	0.331	0.198	1	0.289
(0.5, 0.5)	0.332	0.199	1	0.295
(0.6, 0.5)	0.453	0.294	0.98	0.584
(0.7, 0.5)	0.712	0.593	0.89	0.874
(0.8, 0.5)	0.626	0.81	0.51	0.893
(0.25, 0.6)	0.3	0.176	1	0.181
(0.35, 0.6)	0.309	0.183	1	0.216
(0.45, 0.6)	0.351	0.213	1	0.353
(0.5, 0.6)	0.368	0.225	1	0.396
(0.6, 0.6)	0.453	0.294	0.98	0.584
(0.7, 0.6)	0.712	0.593	0.89	0.874
(0.8, 0.6)	0.626	0.81	0.51	0.893
(0.25, 0.7)	0.3	0.177	1	0.182
(0.35, 0.7)	0.316	0.188	1	0.24
(0.45, 0.7)	0.357	0.217	1	0.367
(0.5, 0.7)	0.377	0.232	1	0.419
(0.6, 0.7)	0.573	0.405	0.98	0.744

Cut-Off (W-Prob, Cl-Score)	F-Score	Precision	Recall	Accuracy
(0.7, 0.7)	0.712	0.593	0.89	0.874
(0.8, 0.7)	0.626	0.81	0.51	0.893
(0.25, 0.8)	0.301	0.177	1	0.186
(0.35, 0.8)	0.322	0.192	1	0.26
(0.45, 0.8)	0.382	0.236	1	0.433
(0.5, 0.8)	0.403	0.253	1	0.481
(0.6, 0.8)	0.603	0.436	0.98	0.774
(0.7, 0.8)	***0.8***	***0.821***	***0.78***	***0.932***
(0.8, 0.8)	0.626	0.81	0.51	0.893
(0.25, 0.9)	0.303	0.179	1	0.195
(0.35, 0.9)	0.34	0.204	1	0.318
(0.45, 0.9)	0.42	0.266	1	0.516
(0.5, 0.9)	0.442	0.284	1	0.558
(0.6, 0.9)	0.717	0.576	0.95	0.868
(0.7, 0.9)	0.796	0.889	0.72	0.935
(0.8, 0.9)	0.455	0.937	0.3	0.874
(0.25, 0.95)	0.307	0.181	1	0.209
(0.35, 0.95)	0.351	0.213	1	0.351
(0.45, 0.95)	0.447	0.288	1	0.567
(0.5, 0.95)	0.483	0.318	1	0.625
(0.6, 0.95)	0.766	0.667	0.9	0.904
(0.7, 0.95)	0.72	0.922	0.59	0.919
(0.8, 0.95)	0.455	0.937	0.3	0.874

Figure 16 shows that among the 40,516 crashes, 2177 (5.37%) have a classification score that is higher than the selected cut-off value (0.80) and can be considered a possible secondary crash. Among the 2,177 crashes, 287 were flagged as secondary crashes by police officers in the crash data, indicating that the model was well trained. In addition, the tool found another 1,890 probable secondary crash candidates.

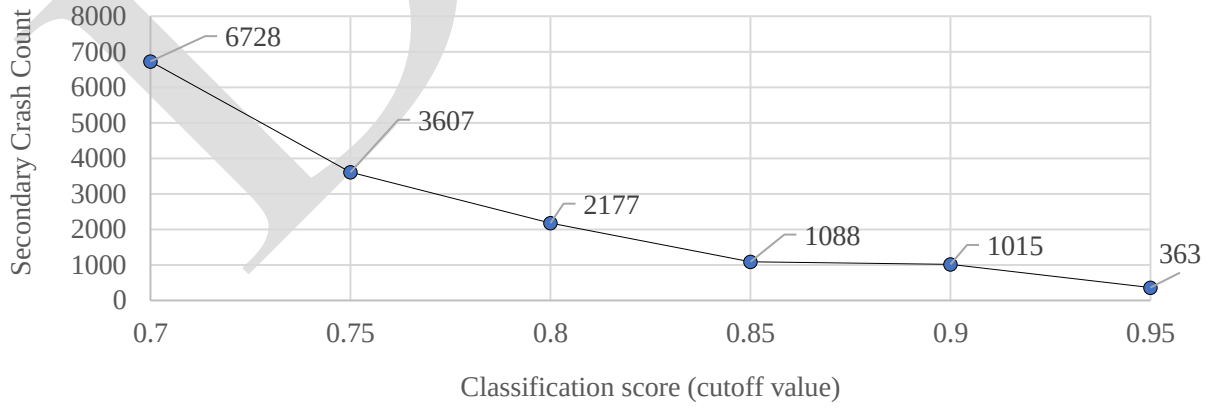


Figure 16: Cumulative Curve with Number of Secondary Crashes vs. the Cut-off Value of the Classification Score from the Test Results for Classifying Secondary Crashes (570 Training Data)

Case Study with 5545 training dataset, 1042 positive cases

For the larger dataset, the model in CIEACT was trained on a DELL laptop that had Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz 1.99 GHz processor, 16.00GB of RAM and 64-bit windows operating system. Model training took approximately 10 seconds. Table 4 shows the top 100 unigrams and bigrams from the trained model, along with their probability scores.

Table 4: Top 100 Unigrams and Bigrams by Probability Score from Model Training

Unigrams	Probability	Bigrams	Probability
Backup	**0.93**	**a previous**	**0.98**
Secondary	**0.92**	**previous crash**	**0.97**
Closure	0.92	94 in	0.96
Primary	0.91	**a secondary**	**0.95**
Cruiser	0.9	*median wall*	*0.95*
Hazmat	0.9	on i39	0.94
Mp	0.9	up due	0.94
Previous	**0.88**	*median shoulder*	*0.94*
i41	0.87	secondary crash	0.94
Congested	0.87	crash ahead	0.94
None	0.86	39 90	0.93
Barriers	0.85	another crash	0.93
Propelled	0.85	of i	0.93
Interstate	0.84	on interstate	0.92
i39	0.84	i39 90	0.91
39	0.83	traffic slowed	0.91
Ahead	0.83	the primary	0.91
Congestion	0.82	of me	0.91
Distress	0.82	on i41	0.91
Assisting	0.81	43 in	0.91
Marker	0.81	*traffic congestion*	*0.91*
41	0.8	traffic was	0.91
Crashes	0.8	41 at	0.9
i94	0.79	separate crash	0.9
Document	0.79	slowing due	0.9
Unrelated	0.79	lane traffic	0.9
Original	0.79	up traffic	0.9
Government	0.78	lane 3	0.9
Temporary	0.78	5 unit	0.89
Blocking	0.78	i 39	0.89
Dep	0.78	just occurred	0.89
Board	0.76	when traffic	0.89
Hudson	0.76	struck no	0.89
85	0.76	other crash	0.89
i90	0.76	another accident	0.89
u3	0.76	to lane	0.88
Reaction	0.76	41 in	0.88
43	0.76	median distress	0.88
94	0.76	i hit	0.88
Reduced	0.74	traffic began	0.88
Tows	0.74	41 unit	0.88
v3	0.74	property was	0.88
Cement	0.74	lanes 2	0.88
90	0.73	chain reaction	0.88
Median	0.73	at mp	0.88
Barrier	0.72	behind upon	0.88

Unigrams	Probability	Bigrams	Probability
Everyone	0.72	3 came	0.88
2019	0.72	1 semi	0.88
Winnebago	0.72	mile marker	0.88
Highland	0.72	distress lane	0.88
Slowing	0.72	unit 5	0.88
Milepost	0.71	ahead unit	0.88
Slowed	0.7	3 vehicle	0.88
Debris	0.7	the disabled	0.88
Semi	0.69	cable barrier	0.88
Separate	0.69	median ditch	0.88
Mile	0.69	lane 4	0.87
2018	0.68	median barrier	0.87
Evasive	0.68	43 at	0.87
Quick	0.68	emergency vehicles	0.87
Disabled	0.68	41 near	0.87
Initial	0.68	multiple vehicle	0.87
Cable	0.68	occurred northbound	0.87
Units	0.67	sudden stop	0.87
Slow	0.67	near mile	0.86
Closed	0.67	slowed due	0.86
Pushing	0.67	got hit	0.86
Blocked	0.67	with emergency	0.86
Overpass	0.67	semi unit	0.86
3	0.66	i94 at	0.86
5	0.66	crash had	0.86
Wall	0.66	was blocking	0.86
Braked	0.66	that crash	0.86
4	0.65	crash north	0.86
Concrete	0.65	construction zone	0.86
145	0.65	43 southbound	0.86
Remain	0.65	i94 in	0.86
Sq	0.65	go traffic	0.86
Emergency	0.64	up ahead	0.86
Rapidly	0.64	state patrol	0.86
0	0.64	happened in	0.86
Braking	0.64	on i	0.86
Abruptly	0.64	traffic ahead	0.86
Visibility	0.63	right distress	0.85
Construction	0.63	interstate 39	0.85
Knifed	0.62	i tried	0.85
Slammed	0.61	was ahead	0.85
Ryan	0.61	middle lane	0.85
Occurred	0.61	license she	0.85
Final	0.61	interstate unit	0.85
Sure	0.6	witnesses upon	0.85
Perpendicular	0.6	4 then	0.85
Sudden	0.6	via radio	0.85
Minutes	0.6	ahead of	0.85
441	0.6	3 stated	0.84
53	0.6	was backed	0.84
Chain	0.6	lane i	0.84
Hard	0.6	the prior	0.84
Avoid	0.6	car behind	0.84
Pushed	0.59	slow in	0.84

Compared to the previous model, this model discovered more unigrams and bigrams related to secondary crashes. In Table 4, the unigrams 'secondary' and 'previous' were related to secondary crashes with higher probability scores (probability score more than 0.87). The top bigrams found in the training related to secondary crashes were 'a previous', 'previous crash', 'a secondary', 'secondary crash', and 'crash ahead' (all having a probability score of more than 0.93). This observation suggests that the classification scores of the narratives and the probability values of the words greater than or equal to 0.95 will be an appropriate cut-off value for identifying secondary crashes from narratives. Many other top 100 unigrams and bigrams shown in the table, such as 'median wall', 'median shoulder', and 'traffic congestion' are not related to secondary crashes but show a very high probability score. These words are commonly found in all types of crash narratives and can affect the classification score.

The model training summary provides different cut-off values for word probability and classification score and are shown in Table 5 below. According to the table, the best results were found with a cut-off value of 0.8 for word probability and 0.95 for classification score. Therefore, in the testing phase a cut-off value of 0.8 was used for word probability and a value of 0.95 was used to classify a secondary crash.

Table 5: Train Model Statistics with Different Cut-off Values for Word Probability (W-Prob) and Classification Score (Cl-Score)

Cut-Off (Wprob, Cl-Score)	F-Score	Precision	Recall	Accuracy
(0.35, 0.5)	0.335	0.201	0.999	0.253
(0.45, 0.5)	0.367	0.225	0.997	0.354
(0.5, 0.5)	0.375	0.231	0.996	0.375
(0.6, 0.5)	0.476	0.313	0.992	0.589
(0.7, 0.5)	0.61	0.445	0.972	0.767
(0.8, 0.5)	0.759	0.655	0.902	0.892
(0.9, 0.5)	0.726	0.9	0.607	0.914
(0.95, 0.5)	0.235	0.986	0.133	0.837
(0.35, 0.6)	0.335	0.202	0.998	0.256
(0.45, 0.6)	0.381	0.236	0.996	0.392
(0.5, 0.6)	0.411	0.259	0.994	0.465
(0.6, 0.6)	0.476	0.313	0.992	0.589
(0.7, 0.6)	0.61	0.445	0.972	0.767
(0.8, 0.6)	0.759	0.655	0.902	0.892
(0.9, 0.6)	0.726	0.9	0.607	0.914
(0.95, 0.6)	0.235	0.986	0.133	0.837
(0.35, 0.7)	0.344	0.208	0.997	0.286
(0.45, 0.7)	0.385	0.239	0.996	0.402
(0.5, 0.7)	0.421	0.267	0.994	0.486
(0.6, 0.7)	0.543	0.375	0.986	0.688
(0.7, 0.7)	0.61	0.445	0.972	0.767
(0.8, 0.7)	0.759	0.655	0.902	0.892
(0.9, 0.7)	0.726	0.9	0.607	0.914
(0.95, 0.7)	0.235	0.986	0.133	0.837
(0.35, 0.8)	0.356	0.216	0.997	0.321
(0.45, 0.8)	0.414	0.261	0.995	0.47
(0.5, 0.8)	0.45	0.291	0.994	0.543
(0.6, 0.8)	0.566	0.397	0.982	0.717
(0.7, 0.8)	0.707	0.561	0.954	0.851
(0.8, 0.8)	0.759	0.655	0.902	0.892

Cut-Off (Wprob, Cl-Score)	F-Score	Precision	Recall	Accuracy
(0.9, 0.8)	0.726	0.9	0.607	0.914
(0.95, 0.8)	0.235	0.986	0.133	0.837
(0.35, 0.9)	0.375	0.231	0.995	0.376
(0.45, 0.9)	0.447	0.289	0.992	0.539
(0.5, 0.9)	0.487	0.323	0.989	0.608
(0.6, 0.9)	0.617	0.451	0.974	0.773
(0.7, 0.9)	0.73	0.594	0.948	0.868
(0.8, 0.9)	0.813	0.808	0.818	0.929
(0.9, 0.9)	0.726	0.9	0.607	0.914
(0.95, 0.9)	0.235	0.986	0.133	0.837
(0.35, 0.95)	0.397	0.248	0.994	0.433
(0.45, 0.95)	0.478	0.315	0.989	0.594
(0.5, 0.95)	0.527	0.359	0.986	0.667
(0.6, 0.95)	0.652	0.492	0.965	0.806
(0.7, 0.95)	0.781	0.674	0.929	0.902
(0.8, 0.95)	***0.813***	***0.822***	***0.804***	***0.931***
(0.9, 0.95)	0.555	0.973	0.388	0.883
(0.95, 0.95)	0.235	0.986	0.133	0.837
(0.35, 0.97)	0.414	0.262	0.993	0.472
(0.45, 0.97)	0.499	0.334	0.987	0.628
(0.5, 0.97)	0.554	0.386	0.983	0.703
(0.6, 0.97)	0.697	0.548	0.959	0.843
(0.7, 0.97)	0.801	0.71	0.917	0.914
(0.8, 0.97)	0.814	0.844	0.785	0.932
(0.9, 0.97)	0.547	0.975	0.38	0.882
(0.95, 0.97)	0.186	0.991	0.103	0.831
(0.35, 0.98)	0.424	0.27	0.992	0.494
(0.45, 0.98)	0.519	0.352	0.985	0.657
(0.5, 0.98)	0.576	0.408	0.98	0.729
(0.6, 0.98)	0.715	0.572	0.952	0.857
(0.7, 0.98)	0.809	0.729	0.91	0.919
(0.8, 0.98)	0.816	0.88	0.761	0.936
(0.9, 0.98)	0.529	0.974	0.363	0.878
(0.95, 0.98)	0.129	0.986	0.069	0.825

The tool took approximately 10 seconds to upload test data and run the model. In Figure 17, the results show that among 40,516 crashes, 4,855 (11.98.%) have a classification score that is higher than the selected cut-off value, so those were not considered to be possible secondary crashes. This model flagged 287 secondary crashes out of 4,855, the same as the previous model. The tool found another 4,568 probable secondary crashes that can be further investigated. As aforementioned, this model discovered more secondary crash-related unigrams and bigrams, resulting in an increase in the number of probable secondary crashes when compared to the previous model.

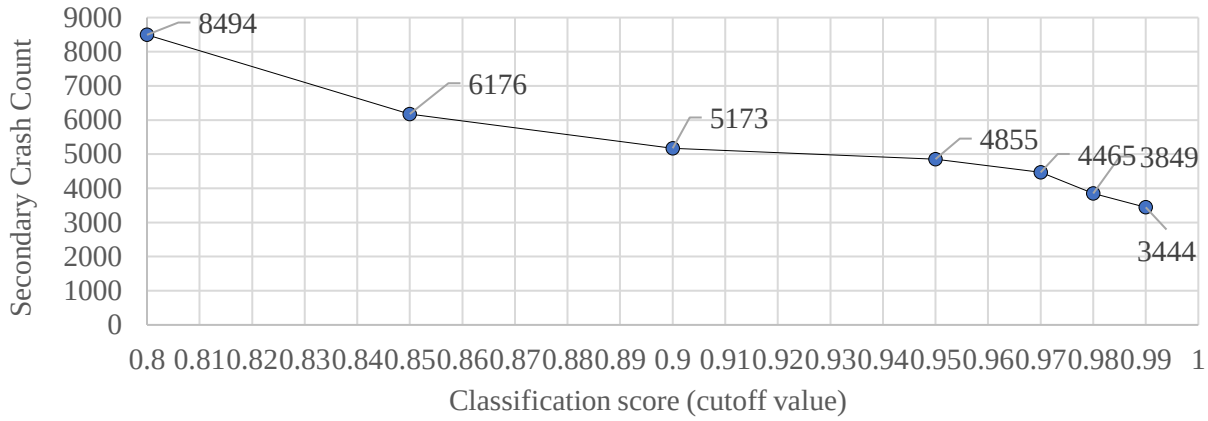


Figure 17: Cumulative Curve with Number of Secondary Crashes vs. the Cut-off Value of the Classification Score from the Test Results for Classifying Secondary Crashes (5,545 Training Data)

The case study with a larger dataset revealed that the tool discovered many unigrams and bigrams related to secondary crashes, resulting in more positive cases than the case study with a smaller dataset. However, due to the presence of many high frequency common words not related to secondary crashes in the training dataset, the test results may produce more false positive cases. Our previous study implies that Noisy-OR works well in at classifying crashes when crash narratives contain more high frequency words related to a particular crash type, such as in the case of distracted driving or work zone crash classification (Sayed et al., 2021). In the narratives of secondary crashes, there were no observations of a narrative containing multiple words related to secondary crashes. In a scenario such as this, the unigram probability and the bigram probability of Noisy-OR can provide valuable insight about the narrative and help us determine whether or not to use the Noisy-OR classifier. If there are few words with high probability values, only unigrams and bigrams will be able to classify the narratives. If the probability value of significant unigrams and bigrams is low, such as 0.70 or 0.75, and the narrative contains very few positive words, it will be difficult for Noisy-OR to retrieve true positives from the narrative.

7. CONCLUSIONS AND RECOMMENDATIONS

The purpose of this project was to develop CIEACT, an online web tool capable of analyzing and extracting valuable information from crash narratives using text mining techniques. The tool is based on the Django web-framework that uses pythons for scripting and provides easy integration of HTML and a relational database. CIEACT provides an opportunity to develop multiple models concurrently and analyze the results. CIEACT was tested both offline and online using different types of electronic devices. During the testing process, the tool showed satisfactory performance with all designed functionalities working properly. The authors also conducted a case study to evaluate the functionality of a custom model that allows users to train their model in real time. The tool showed satisfactory performance in classifying secondary crashes.

CIEACT can assist safety practitioners and professionals in extracting valuable information from narratives, recovering missed crashes, and reviewing the results in both tabular and spatial contexts. Federal and state transportation agencies can use this tool to analyze crashes and implement new policies to enhance structural data quality and determine effective safety measures.

The most attractive aspect of CIEACT is its simplicity and flexibility. The user interface includes detailed instructions, allowing a user to easily navigate to different webpages within the tool. In terms of flexibility, the tool provides both a default built-in model and the ability to train a model in real time for crash analysis and classification. The tool presents the results in tabular format with color coding and a classification score. Finally, users can download the result in a csv file for further review and analysis.

CIEACT gives users the option of training their own models or utilizing a default model that has already been trained. It uses a relational database PostgreSQL to store information from the default model that can be used to update any existing default model or to add a new model. The tool handles the default model at the database level and the user-trained model at the memory level to reduce computational time. The current version of CIEACT uses a Noisy-OR based hybrid (unigram + bigram) classifier for both the default model and the training of a new model. The default model is applicable for ‘wokzone’ crash classification. However, we designed the tool in such a way that its functionality can be scaled up by integrating other text mining algorithms. Future versions of the tool are expected to include additional crash classification models, such as ‘distracted,’ ‘inattentive,’ and ‘secondary,’ as well as different versions of models, such as ‘NoisyOR-unigram,’ ‘NoisyOR-bigram,’ and ‘NoisyOR-trigram.’ Future versions could also include machine-learning algorithms.

ACKNOWLEDGEMENT

This study was supported by a grant from the National Highway Traffic Safety Administration (NHTSA) through the Wisconsin Department of Transportation (WisDOT) Bureau of Transportation Safety (FG-2021-UW-MILWA-05662). Thanks to the University of Wisconsin-Madison Traffic Operations and Safety (TOPS) Laboratory for making crash narrative data available for this research.

REFERENCES

Douglas, K., & Douglas, S. (2003). *PostgreSQL: A Comprehensive Guide to Building, Programming, and Administering PostgresSQL Databases*. Sams Publishing.

FHWA. (2020, October 19). *Traffic Incident Management Performance Measurement Presentation—FHWA Focus States Initiative: Traffic Incident Management Performance Measures Final Report—FHWA Emergency Transportation Operations*. https://ops.fhwa.dot.gov/publications/fhwahop10010/presentation.htm

Oniśko, A., Druzdzel, M. J., & Wasyluk, H. (2001). Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. *International Journal of Approximate Reasoning*, *27*(2), 165–182. https://doi.org/10.1016/S0888-613X(01)00039-1

Sayed, M. A., Qin, X., Kate, R. J., Anisuzzaman, D. M., & Yu, Z. (2021). Identification and analysis of misclassified work-zone crashes using text mining techniques. *Accident Analysis & Prevention*, *159*, 106211. https://doi.org/10.1016/j.aap.2021.106211

Vomlel, J. (2006). Noisy-or classifier. *International Journal of Intelligent Systems*, *21*(3), 381–398. https://doi.org/10.1002/int.20141

Zagorecki, A., & Druzdzel, M. (2004). An Empirical Study of Probability Elicitation under Noisy-OR Assumption. *Flairs Conference*, 880–886.