LLM-Based Predictive Modeling for Traffic Flow Optimization Using Real-Time Social Media Data

PMCA698J - Dissertation-I/ Internship-I

Submitted in partial fulfillment of the requirements for the degree of

Master of Computer Applications

in

Department of Computer Applications

by

Priom Dutta

24MCA0169

Under the guidance of

Dr. Tapan Kumar Das

School of Computer Science Engineering and Information Systems

VIT, Vellore

November, 2025

Executive Summary

The situation of traffic jam, road accident and random flow of vehicle become more and more serious in urban area, thus the management of traffic jam has been a critical problem now. Traditional traffic prediction and route planning methods are based on static infrastructures like sensors, GPS devices and historical traffic flow data. During unplanned events like accidents, public events, or even road closures, these systems are too slow to react and provide the necessary response to such chain events and this affects the actual flow of traffic very much

In the meantime, social media platforms have sprung up as valuable public information resources, where travellers often post real-time information warning of road closures, delays and accidents. Although being unstructured and noisy this data offers the possibility to learn from the extensive contextual information of the road which, when being used appropriately, can help to enhance traditional ways of traffic prediction. But it is not easy to extract useful signals from informal text when engaging challenges in relevance filtering, entity recognition and event classification.

Motivated by this query understanding task, in this study, we propose a method to use Large Language Models (LLMs) to predict what information in the social media posts is associated with traffic. Using natural language processing, the system attempts to predict traffic related incidents with greater context, recognizing events and making sense of textual clues about traffic. The resultant product is a hybrid approach that balances between linguistic intelligence and predictive modeling and produces not only more flexible, but also knowledge-based traffic flow optimization.

Keywords: Traffic flow optimization, Large Language Models (LLMs), Natural Language Processing(NLP), Social Media, Incident Prediction

CHAPTER 1

INTRODUCTION

Traffic jams have continued to pose a challenge in the urban setting causing delays, lost money and environmental consequences. The conventional traffic policing strategies, including fixed sensors, GPS positioning, and video monitoring systems offer useful feedback but might not be timely and as detailed as the real-life information supplied by the population. During the past few years, there has been the advent of social media applications such as X (which was previously Twitter) that have become a source of complementary data, with the users posting real-time updates regarding road conditions, accidents, and traffic jams.

The increasing capacities of Natural Language Processing (NLP), in particular, Large Language Models (LLM), provide an opportunity to handle and comprehend such unstructured text productively. The ability to understand semantically, detect sentiment, and identify location-specific events via twitter allows LLDM to understand a live traffic situation better. These insights are usable as predictive models to facilitate optimisation of traffic flows when paired with both time and space data.

The study hypothesizes a predictive modeling framework that will use the time-sensitive information on social media to analyze traffic in Vellore. The intended methodology involves five steps, namely the formulation of a problem and review of the literature, data collection and preprocessing, feature extraction using the LLM, developing a predictive model, and optimization through visualization. Embedded generation Lightweight LLMs, like BERT or DistilBERT, will be discussed, and then machine learning or deep learning techniques, like LSTM will be used to predict traffic.

Although the research is at the formulation and review stages, a systematic pipeline to predict traffic using tweets and an interactive visualisation dashboard are some of the expected outcomes. In this work, the author attempts to assess the practiceability of applying the LLM to traffic management systems to help in creating smarter and more receptive city mobility solutions.

1.1 Objective

The overall aim of this research outcome is to create, innovate and test a hybrid intelligent system which will help to predict traffic jams based on real time social media data which will be majorly obtained through platforms like X (thus has become twitter) and similar platforms. The system uses semantic text comprehension of text-based data

with the help of Large Language Models (LLM) and predicts traffic situation on an adaptive and data-driven basis with the assistance of temporal deep learning models (LSTM).

The key goals of this work are as follows:

1. To extract the contextual meaning, sentiment, and event information by using state-of-the-art Natural Language Processing (NLP) technologies built around Large Language Models (e.g., all-MiniLM-L6-v2) based on successive net posts to real-time social media regarding traffic scenarios.

2. To filter and arrange unstructured social information, such as cleaning tweets, non-English, as well as irrelevant tweets, and location text geotagging, to feed the models effectively.

3. To generate embedding-based semantic representations of social media texts using lightweight LLMs, ensuring computational efficiency while preserving contextual understanding.

4. To integrate Long Short-Term Memory (LSTM) networks with the embeddings derived by LLM to carry out time-series predictions of traffic congestion levels of traffic, taking into account linguistic and temporal dependencies.

5. To create a hybrid predictive model that is used to classify real-time traffic conditions into a variety of congestion types (e.g., Low, Medium, High) and determines traffic density and delay metrics of individual locations.

6. To train and validate a hybrid model on predictive performance through suitable evaluation metrics (i.e. Accuracy, Precision, Recall, F1-score, Confusion Matrix) and compare it to baseline models of the Machine Learning (i.e. Random Forest and XGBoost).

7. To visualize model predictions using a user-friendly dashboard, allowing one to follow the trends of predicted congestion dynamically, traffic density maps, and estimates of delay so that the predictions can be used to make decisions and improve people awareness.

8. To prove that social-media-based traffic prediction like real-time, economical and scalable, is a feasible alternative to traditional sensor-based Intelligent Transportation Systems (ITS).

These objectives are to create an intelligent, interpretable, and real-time traffic forecasting architecture that is a fusion of the semantic understanding properties of LLMs and the temporal analysis power of LSTM networks that will offer actionable conclusions on optimization of urban mobility and congestion control as final output of the project.

1.2 Motivation

Due to the rapid rise in urbanization and passenger density in the contemporary cities, the cities have become grossly congested, leading to long journeys, economic waste, as well as environmental wastage. Various transport researches indicate that most of the urban commuting time is wastage in case of erratic traffic problems, unplanned road diversion, and insufficient dissemination of information to commuters. The use of Traditional Intelligent Transportation Systems (ITS) though useful in controlled settings is usually based upon infrastructural sensor systems, GPS equipment and video surveillance systems which are very expensive to install and have limited coverage.

On the contrary, the social media platforms like X (previously twitter) have surfaced as real time as well as crowd sourced information streams. The estimates of the number of users who have shareable updates concerning road accidents, road congestions, weather inconveniences, and accidents are in the millions. Improperly filtered and analyzed, these textual reports can serve as real-time, location-dependent updates of the traffic state, producing a huge, constantly changing stream of information which cannot be matched on a large scale by traditional sensor systems.

Nevertheless, the amount of social media data is not structured and noisy, making it difficult to analyze. Tweets can be rather brief, casual, add full of slang, emojis, and unclear situations. Ordinary machine learning models are incapable of expressing this kind of linguistic subtleties. Large Language Models (LLMs) here present transformational opportunities since they know how to use context and labeling to elicit understanding. LLAMs can extract finer details of textual information by representing it in meaningful numeric representations, including purpose, emotion and the extent of

traffic conditions.

Moreover, traffic behaviour is always temporal and sequential- there are changes of the level of congestion within different time periods and have to be shaped by previous traffic dynamics. Long Short-Term Memory (LSTM) networks can be used to successfully predict such changes, in which case LLM-based embeddings are combined with Long Short-Term Memory networks in a hybrid mode, and both language semantics and time-dependent changes are processed at the same time.

The driving force of this work, thus, can be seen in closing this divide between the information flows of social media and clever traffic forecasting. The suggested hybrid LLM-LSTM model could become a cheaper, versatile, and real-time traffic analyzing framework, which can continuously learn on the basis of the crowd-sourced data with no expensive physical sensors required. Such a strategy helps to not only make the traffic information more accessible, but also leads to the realization of sustainable city mobility, intelligent urban infrastructure, and data-driven civic safety networks.

Academically / technically, the proposed project is the possibility to apply innovative ideas of Artificial Intelligence (AI), in the field of NLP and Deep Learning, to real-world urban issues. It demonstrates that the contextual language comprehension (LLM) and sequence models (LSTM) in a combination, when used separately, can present actionable information to maximize traffic flow and minimize delays and improve commuter experiences in the real-life context.

1.3 Background

The Intelligent Transportation Systems (ITS) has always been a research field in which traffic management and optimization have been key issues. The real-time traffic monitoring, congestion forecasting, and incident detection are necessities which have become a necessity as the cities keep expanding in population and vehicular density. Conventional ITS systems are based on a network of cameras, loop detectors, radar detections, and GPS sensors that are used to measure the movement and flow statistics of vehicles. These systems have been effective in the sophisticated urban infrastructure environment, but they are characterized by high initial cost of set up, low scalability, as well as low geographical coverage.

Simultaneously, the fast-growing of social media has changed the mechanisms of producing and exchanging information. There are sites like X (Twitter), Facebook, and Reddit, where technologies create a vast flow of user-created textual information within a second, and the subject of the posts may be weather conditions, road accidents, and crises. That is why social media plays the role of a shared sensor network with users having the option of reporting real-time occurrences by themselves. The use of this data to drive traffic intelligence has become a potential area in the research in recent times within the umbrella concept of Social Sensing and Crowd-Sourced Traffic Analysis.

Nevertheless, using the social media data to make its predictions on the traffic creates a number of challenges. It is unstructured, and it is written in natural language and it is usually mixed with irrelevant or sarcastic information. Traditional text-mining and keyword-based methods cannot be sufficient to grasp the contextualized meaning and the semantic association contained in human language. As a result, attention to research has been drawn to the Natural Language Processing (NLP) and Deep Learning tools that can comprehend more complicated linguistic structures.

1.3.1 Natural Language Processing and Embeddings

The creation of word embedding models Word2Vec, GloVe, and FastText, has created the basis of machine understanding of textual data. The models converted words into numerical vectors, and the algorithms were trained to acquire such relationships as similarity and sentiment. The previous embeddings were however context free with a word being represented identically in every sentence that it occurred in. This was a weakness that limited the application in practical scenarios such as traffic analysis where the interpretation of words such as jam, clear, or blocked varies in every situation.

Use of contextual embedding Transformer-based Large Language Models (LLMs), including BERT, DistilBERT, and MiniLM, have now revolutionized the study of NLP. These models perceive the semantics of complete sentences and not of individual words and are used to represent more linguistic and syntactic dependencies. Nowadays, the LLLMs can recognize the smallest details, including whether the tweet is frustration, relief, or sarcasm, which makes them a great fit to interpret the social media-related traffic information.

1.3.2 Temporal Nature of Traffic Data

Traffic patterns are arrived at with time dependencies. Due to the different human mobility and environmental conditions, congestion varies among the hours, days and seasons. Analysis of these time variations can only be made possible through algorithms capable of learning, as a result of sequential data. The original time series data architecture was Recurrent Neural Networks (RNNs), which were affected by the vanishing gradients and the short-term memory issue. The limitations of Long Short-term Memory (LSTM) networks were introduced to address these limitations, which entailed the inclusion of memory gates through the prospect to choose what to remember and forget of the past information. This renders LSTM specifically suitable in predicting time-dependent trends like traffic congestion, vehicle flow, and delay trends.

1.3.3 Deep Learning in Traffic Prediction (Hybrid)

Recent innovations in the hybrid deep learning architectures have revealed that hybrid combinations of one type and another can considerably boost prediction. As an example, semantic information extracted by LLM based embeddings upon input of text by means of embedding and fed to an LSTM model enable one to understand what is being said (semantic context) and when it is said (temporal context) at the same time. It is also a suitable type of hybrid LLM-LSTM architecture, as in the context of predicting traffic (and other real-world scenarios), the content (e.g., "accident near Katpadi road) and the timing (e.g., "at 8:30 AM) are equally critical.

1.3.4 Social Data Real-Time as an Accessory Traffic Sensor

The combination of real-time and crowd-sourced information of such services as Twitter provides a chance to complement the use of traditional traffic sensors. Twitter messages about the state of the roads or accidents or any kind of congestion can deliver immediate feedback even in places where physical sensors are inaccessible. In addition, this type of approach is rather inexpensive, scalable, and adaptable, as the source of data under it (social media activity) is constantly being changed without any further investment into the infrastructure

The problem is that it is not always easy to separate relevant, location-specific, and contextually accurate tweets with an enormous stream of information. This requires strong data preprocessing, filtering and feature extraction mechanisms all which are accomplished in this study through cleaning pipelines, geotaggings and semantic embeddings.

1.3.5 Proposal Work Positioning

Based on these advancements, the current study combines the implementation of LLM in understanding the text and LSTM in prediction of time series to construct an integrated traffic prediction system. The model predicts traffic jam of Low, Medium and High levels by extracting embeddings with all-MiniLM-L6-v2 which is a compact though powerful sentence transformer and then through a combination of embeddings and temporal data (timestamps and locations). It also approximates the roadway traffic and the potential delay times of certain areas.

This approach is unlike the traditional systems, which rely on sensor networks, but it shows how data-driven AI models may use social media as a virtual sensor network, and in effect, the linguistic data can be converted to actionable traffic information.

Literature Survey :

Table 1.1 Literature Survey

S.NO	TITLE	MERITS	DEMERITS
1	Towards explainable traffic flow prediction with large language models	The article is also valuable because it explicitly presents the critical issue of explaining traffic prediction AI. It presents a novel and significantly innovative methodology, which transforms intricate traffic signal data into natural language so that a Large Language Model will provide natural language explanations of its predictions.	The article has not discussed much on practical implementation and scalability. Data quality is vital to the performance of the paper, and may be overfitting, and is computationally very complex, potentially becoming impractical in resource-constrained settings.
2	Traffic flow prediction for	This paper analysed several machine learning (ML) and	The article has used the data that consists of 56

	smart traffic lights using machine learning algorithms	deep learning (DL) models to predict traffic intersection to optimize adaptive traffic lights systems. Multilayer Perceptron Neural Network (MLP-NN) had the greatest performance and its R-squared and explained variance value was 0.93 and this indicates its possibility in real-time application in traffic management.	days that might not be able to reflect the seasonal changes in traffic. Also, as much as the research is on the prediction of traffic flow, it fails to consider dynamic adjustment of the timings of the traffic lights in line with predictions, which restricts its use in adaptive traffic control systems.
3	Big data analytics in intelligent transportation systems: A survey	The article will effectively offer a summary of the big data analytics in Intelligent Transportation Systems (ITS), identifying the different sources of data, processing methods and their uses. The article demonstrates the promise of the big data to improve traffic management, safety, and efficiency as one of the key sources of information to be used by scholars and professionals in the area.	The article mainly employs the theoretical part and is not followed by descriptions of specific cases or real-life practices. The paper fails to mention the issues associated with data privacy, security and integration amongst various transportation system which are important in the real world application.
4	Real time traffic prediction based on social media text data using deep learning	The article is a live traffic forecast model based on the usage of social media textual data i.e. Twitter that are processed using the framework of Spark and Kafka. The article adopts a scheme of ensemble neural network to improve the prediction quality and come up with a scalable dynamic traffic management scheme.	The model can be constrained by this because the paper utilizes Twitter data and therefore cannot be applied to other areas or platform that share similarity about the user behaviors. The possible issues pertaining to the privacy of data and incorporation of this model into the already used traffic management systems have also not been tackled in the paper.

5	Traffic prediction using time-space diagram: a convolutional neural network approach	The article presents a deep learning-based algorithm, based on Convolutional Neural Networks (CNNs), of the direct prediction of traffic conditions in the form of time-space diagrams based on the data of connected vehicles. The paper has proven that CNNs are more effective in predicting traffic movements and density than conventional models such as Multilayer Perceptron, Support Vector Regression and ARIMA.	The complexity of CNNs may force the paper to consume large amounts of computational resources. This paper also considers that connected vehicle data are readily available, which is not as common across all areas, and that could restrict the application of the model.
6	Artificial intelligence-based traffic flow prediction: A comprehensive review	The article gives a detailed discussion of machine learning and deep learning approaches used in the prediction of traffic flow. It provides valuable knowledge to researchers in the field as well as practitioners because the paper reveals internal challenges to the implementation of these methods in predicting traffic.	The article mainly employs the theoretical part and is not followed by descriptions of specific cases or real-life practices. The paper fails to mention the issues associated with data privacy, security and integration amongst various transportation system which are important in the real world application.
7	Spatial-temporal graph sandwich transformer for traffic flow forecasting	The article proposes a new architecture called Spatial-Temporal Graph Sandwich Transformer (STGST), which is used to forecast traffic flow. In the STGST, two time encoding Transformers, a time encoding spatial Transformer, and one structure and spatial encoding Transformer are utilized with a sandwich configuration. The design has a good capacity to capture long-range temporal and deep spatial dependencies, and makes the model more inclined to become familiar with the intricacies of traffic patterns.	The architecture of the paper can be very computing intensive as it exploits the Transformer architecture. Although it works fairly well on benchmark datasets, its applicability in dynamic real world traffic situations where data quality changes is still unknown. There are also concerns of privacy and security of the data as well as its integration with current traffic systems that are yet to be discussed.

8	Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting	This paper presents a new deep learning model called STGCNs, which is aimed at traffic prediction. The model can identify both spatial and temporal relations of traffic data by using graph convolutional layers and convolutional sequence learning. The article illustrates that the STGCNs are more effective than traditional techniques, which require more time to train and also, they are capable of providing higher accuracy on practical datasets .	The model itself can be rather difficult, which means that the paper will have to consume considerable resources through computation. Also, their weights on graph-based representation presuppose the presence of in-depth road network data, which is not always available in unusual areas, and it can be a limitation to the application of the model.
9.	Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting	This paper describes a deep learning architecture, the Traffic Graph Convolutional Recurrent Neural Network (TGC-RNN) that can be used to predict traffic on a scale of a network. This model integrates rapidly spatial dependencies of the traffic networks with temporal dynamics in the traffic flow through the combination of the graph convolutional networks (GCNs) and the recurrent neural networks (RNNs). The article has established that TGC-RNN is better than traditional models, has higher accuracy, and interpretability in traffic prediction tasks.	The complexity of the combined GCN and RNN architecture may demand important computation resources, which is the approach used in the paper. Moreover, the model could be the least applicable in regions with a low density of traffic networks or incomplete data because it requires detailed road network data.
10	Large language models (llms) as traffic control systems at urban intersections: A new paradigm	The article suggests applying Large Language Models (LLMs) to control urban intersection traffic, which employs the reasoning and decision-making capacities of the models. According to simulations, the accuracy is high (83%), and the processing performance is good to identify and assign priorities to traffic and optimize waiting time, which may help to push the rating	The article uses simulations and this might not reflect the variability in traffic in the real world. The computational requirements are high in LLMs, and implementing them in current infrastructure is problematic regarding the privacy and security of the data and compatibility with the

		of the traffic management among LLMs up.	systems.
11	Enhancement of traffic forecasting through graph neural network-based information fusion techniques	The article presents a new concept of traffic prediction involving the use of Graph Neural Networks (GNNs) in combination with information fusion methods. This mixed model is able to well represent multifaceted spatial and time-based relationships in traffic data resulting in a greater forecasting error. The paper shows that the model is better than the traditional methods by showing the results of thorough experiments on real-world datasets, which reveals that the model can be applied to the real-life management and planning of traffic.	The complexity of the integrated GNN model can be critical to the calculations that are necessary in the approach of the paper. Moreover, the model will require high-quality data which may restrict its application in the areas with low or sporadic traffic data. The challenges connected with the large-scale urban network implementation of the model are also not discussed in the paper.
12	Traffic information mining from social media based on the MC-LSTM-Conv model	The paper proposes a new model, called MC-LSTM-Conv, which integrates Multi-Channel Long Short-Term Memory (MC-LSTM) networks with convolutional layers in order to extract the information about the traffic on social media. The hybrid model is also able to capture spatial and temporal relationships in traffic data making it more accurate in predicting traffic flow. The article proves the effectiveness of the model using experiments on real-life data and demonstrates that this model can be used in real-time traffic monitoring and control.	The method adopted in the paper can be resource-intensive (on the compute) because of the complexity of the MC-LSTM-Conv model. Moreover, the usage of social media data can bring noise and inconsistencies which would impact the robustness of the model. Another issue that has not been discussed in the paper is the problems associated with privacy and security of data when others use socially media materials released publicly.
13	Traffic Flow Prediction Based on Large	The article suggests R2T-LLM, a type of using large language models to	The paper has a high demand of computational resources

	Language Models and Future Development Directions	transform multimodal traffic data into explainable forecasts. It effectively captures the spatiotemporal trends, matches deep learning models in terms of accuracy, and provides the information useful in predicting the future traffic conditionally.	and depends on intense data preprocess and is not easy to integrate with the current traffic systems in terms of privacy and compatibility.
14	Embracing large language models in traffic flow forecasting	This paper presents LEAF, a new traffic flow predictor model which incorporates two streams of expression of relationships between space and time in the form of graph and hypergraph. In the process of inference, LEAF uses a large language model to choose the most likely prediction between the two branches. It increases the flexibility to dynamic traffic environments and accuracy of forecasting.	The computational resources required to comply with the vast language models can be very large. Furthermore, the efficiency of the model in the scenario of real high-noise traffic conditions is also yet to be tested in failure. Some of the challenges that may be encountered with the implementation of LEAF in the current traffic management systems are that of data privacy, security, and compatibility of the systems.
15	Traffic Detection and Forecasting from Social Media Data Using a Deep Learning-Based Model, Linguistic Knowledge, Large Language Models, and Knowledge Graphs	The paper suggests a three-phase model based on deep learning, NLP, LLM, and knowledge graphs to identify and predict traffic on the basis of social media data. It improves accuracy, semantic and temporal-spatial intelligent transport systems reasoning.	It is computationally intensive and based on noisy social media data, which indicates difficulty in implementing it in practice in terms of privacy, quality of data and information.

CHAPTER 2

DISSERTATION DESCRIPTION AND GOALS

2.1 Description

This dissertation is devoted to the creation of a smart, hybridized predictive model, which will combine Large Language Models (LLMs) and Long Short-Term Memory (LSTM) networks to optimize the traffic flow based on real-time data on social media. With congestion on the roads becoming a regular reality of our contemporary city life, there is an urgent necessity to develop the system capable of understanding the human-generated online information and using it to obtain the instance of the actionable data in the form of the traffic. The proposed study is intended to fulfill this requirement, creating a hybrid structure of learning that can process unstructured social media text and make predictions about the indexes of congestion and potential traffic congestion in a structured and real-time manner.

The suggested work uses a semantic level-based approach of deriving textual social data i.e. tweets that can carry great situational information regarding road conditions, accident, or traffic jams. Tweets are unsynchronized, informal and context sensitive and therefore the conventional methods of text-mining are useless. Thus, the embedding model, all-MiniLM-L6-v2, based on transformers, has been used to encode every tweet in a numeric form that represents the linguistic meaning and the dependencies in the context. These embeddings are presented as semantically rich input features that are fed to a temporal deep learning network (LSTM) which learns sequential dependencies in the data. The proposed hybrid predictive model of the LLM + LSTM is based on the combination of these two technologies.

The project workflow has a systematic and research-oriented approach, which is applied in five separate phases:

The initial stage, entails lit survey and the problem formulation. Within this phase, the gap in the research was narrowed down by researching on the currently available technologies to predict traffic through marking machine learning and sensor technologies. The purpose of using social media data, especially the X (formerly Twitter) was determined as a viable and low cost alternative to traditional ITS systems.

The second step, involves data gathering and pre-treatment. The twitter feed was gathered in real time via the interface of Tweepy and Rapid API, filtered with keywords related to traffic, and cleaned by eliminating noise data like hash tags, links, and special characters. Other metadata like timestamps, positions of the users and interactions of tweets (likes and retweets) were also snatched out. The data was then converted into a tabular form that was to be fed in the model training, the columns included the twit id, createdat, rawtexttweet, userlocation, retweetcount, and likecount.

The third phase was carried out to run LLM embedding generation and feature engineering. All of the tweets were coded into a dense embedding with all-MiniLM-L6-v2 transformer. These embeddings were then clustered using HDBSCAN and K-Means in order to classify similar traffic-related situations in the process of creating unsupervised congestion clusters. Such embeddings were also included with temporal features, including time of day and day of the week and location coordinates, producing a very detailed feature matrix, which both represents textual and temporal features.

The fourth stage, is devoted to the development and testing of the predictive model. The hybrid model is an integration of the LLM embeddings together with LSTM network to forecast the categorical congestion levels (Low, Medium, and High). Random Forest and XGBoost were also used as other baseline models to prove that the models would perform comparatively. The hybrid model had a significantly good accuracy ([?]99.6) on the validation data, meaning that the generalization in classifying the level of traffic congestion is very high. Different performance measures including Accuracy, Precision Recall and F1-score were calculated to determine the reliability of the models used. The model was also generalized to predict the traffic density and the anticipated period of delay of certain places according to the anticipated congestion forecast..

Lastly, the fifth stage is dedicated to optimization of the system, its visualization, and implementation. Hyperparameters of the model had been optimized to give them the best model to reduce overfitting and enhance stability. A friendly Streamlit dashboard was created to allow the physical visualization of an outcome. There are several features incorporated into the dashboard- live data on congestion prediction by tweets, bulk CSV uploads to analyze lots and lots of data, heatmap visualization of congestion, and estimation of delays in an easily-readable format. It has a user-friendly intuitive

interface that enables users and stakeholders to track the expected traffic.

In general, the dissertation adds a novel AI-based framework proving the practical opportunities of the social-media-based traffic prediction. This model is a data-driven, scalable, and cost-effective alternative to a traditional sensor-dependent software, and can dynamically respond to real-world events due to online social data streams. The use of LLM embeddings and LSTM part help to provide a thorough contextual interpretation of language and understand the sequential and time-varying character of traffic data, respectively. This union fills the void between semantic text knowledge and time prediction that posits the system as being intelligent and adaptive. The prototype hybrid model with an interactive dashboard can serve as a successful prototype in further smart city traffic management systems and help improve the evolution of smart city mobility and intelligent transportation analytics.

2.2 Goals

The key objective of the given dissertation is to design and execute an intelligent, data-driven hybrid predictive model that takes advantage of the joint forces of Large Language Models (LLMs) and Long Short-Term Memory (LSTM) to forecast the level of traffic congestion through social media in the real-time scenario using real-time social media information. The study will focus on showing how informal, people generated information like tweets can be utilized to generate significant insights that would streamline the traffic movement in cities and improve transportation efficiency.

The general aim of this is to fill the longstanding gap between natural language comprehension and time-series prediction by combining state of the art natural language processing methods and time-series prediction based on deep learning. The system hopes to use crowd-sourced data on applications, such as X (which used to be Twitter) as a virtual sensor network, to thus supplement the conventional traffic monitoring systems with a low-cost, scalable and real-time information source.

To be more exact, the following significant objectives shape the project:

- To examine the usefulness of social media data as a predictive representation of the actual traffic scenario, it is necessary to determine how often user-generated postings can be useful to build predictions. The system will seek to ensure that

traffic-related tweets have enough contextual, temporal and sentiment based data that can be used to make inferences about real-time road conditions.

- To preprocess and format a unstructured textual data that is acquired through a social media source into format that can be used in computational modeling. It comprises cleaning of data, language filtering, geolocation guessing, and sentiment removing to improve the quality and relevance of the dataset that is used to train a model.

- To obtain meaningful feature representations using embeddings by the use of LLM, it is necessary to allow the model to internalize rich semantic insights on the textual material specifying traffic. Employing models of the transformer architecture such as all-MiniLM-L6-v2, the completion attempts to transform the original Twitter tweets into large numerical representations to encode the context, tone, and traffic relevance.

- To come up with a hybrid LLM-LSTM predictive model that integrates both contextual embedding and temporal connections in the accurate prediction of traffic congestions. The LSTM layer takes the patterns of input in a sequence e.g time and frequency of incident whereas the LLM embeddings brings a semantic awareness of the subjects by the tweet corpus.

- To categorize and forecast the level of traffic congestion into specific groups including Low, Medium and High, using the real-time data streams. The purpose is to have a high predictive performance by measuring Accuracy, Precision, Recall, and F1-score of evaluation metrics to provide reliable results in a variety of data situations.

- To estimate the supplementary traffic parameters i.e. traffic density, anticipating delay duration, etc. based on the forecasted congestion types.This improves the usability and interpretability of the output of the model in practice within a traffic management system.

- To test the performance of the hybrid model on top of the baseline machine learning algorithms (Random Forest (RF) and XGBoost), and, therefore, confirm the added value of the introduction of LLM embeddings and the LSTM

architecture.

- To install a convenient visualization GUI that enables end-users can interact with the predictive system in real time. The dashboard in the form of a Streamlit strives to offer the capability of single-tweet prediction, bunches of CSV parsing, heatmap of congestion, delay prediction and downloads of the results - making sophisticated outputs of AI cognizable as straightforward visual concepts.

- To achieve scalability, adaptability and robustness of the model to such a level that it is able to accommodate large amounts of real-time data and be able to adjust to changing trends in traffic without visiting retraints that would require extensive retraining.

All these objectives aim to develop a more intelligent and detailed framework that does not only forecast the pattern of congestion well, but also adds in the direction of real-time all-time decision support of smart city infrastructure. Through integrating linguistic intelligence and temporal learning, the system hopes to revolutionize the way social information will be used in traffic control that will lead to adaptive AI-based transportation.

CHAPTER 3

TECHNICAL SPECIFICATION

3.1 System Requirements

The hybrid model of the traffic prediction system proposed based on using Large Language Models (LLM) and Long Short-Term Memory (LSTM) networks, demands an appropriate environment on a hardware and software layer to guarantee the effective implementation of data processing, embedding generation, training of the model, and visualization. This project was developed through Python frameworks and deep learning libraries through a contemporary computing environment

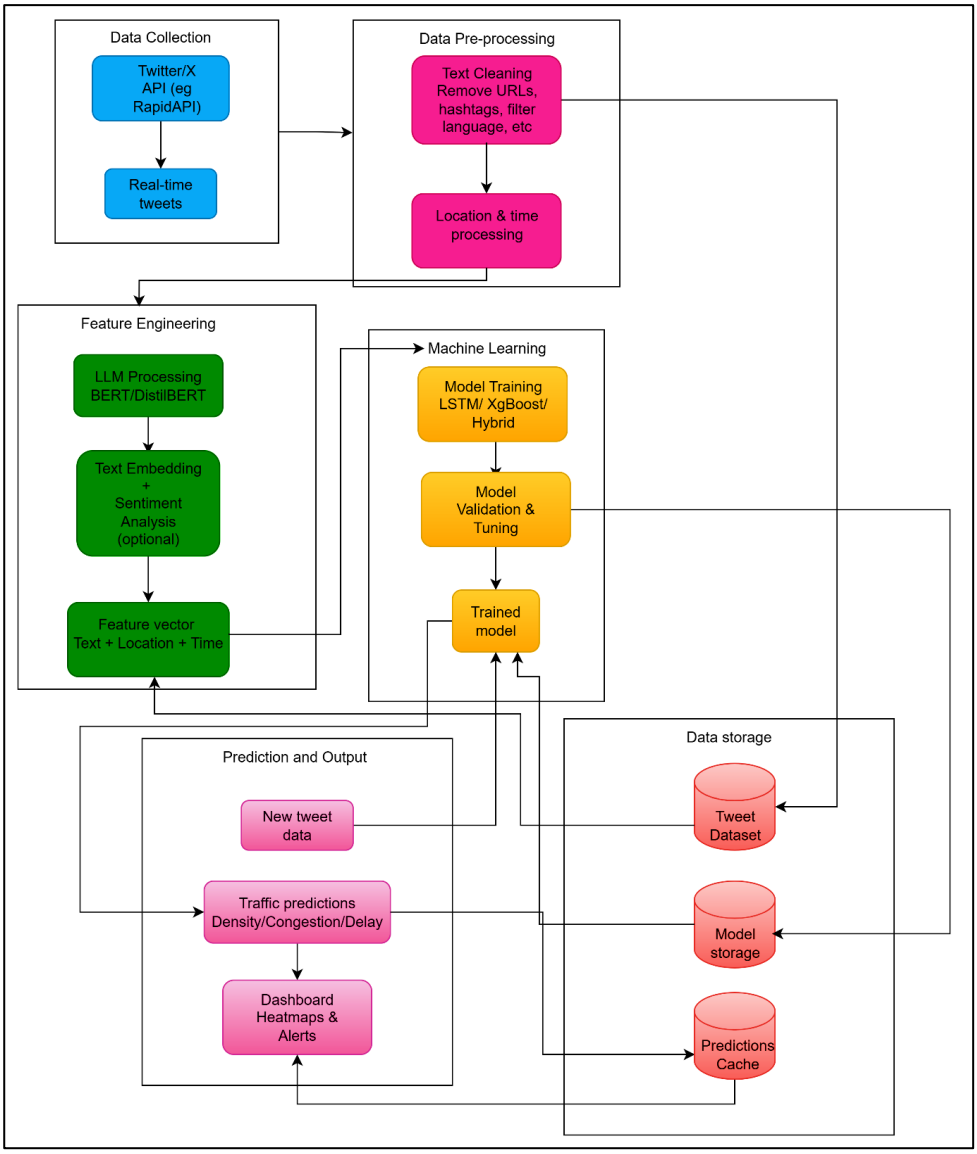
The detailed requirements are as follows:

- Hardware Requirements :
    - CPU : Intel core i5 or above
    - RAM 8 GB (16 GB suggested in faster model inference)
    - Storage : 25 GB of minimum free storage space of datasets, embeddings, and model weights
    - GPU(optional) (CUDA-compatible) through NVIDIA in order to run the models faster.
    - Operating System Windows 10 / 11 or Ubuntu 20.04+
- Software Requirements :
    - Programming Language : Python 3.10 or higher - selected due to its wide support of data science, natural language processing, and deep learning.
    - Development Environment : VSCode
    - Frameworks & Libraries :

- ✓ Tensorflow/Keras for building, training and optimizing the LSTM deep learning model.
- ✓ Scikit-learn for data preprocessing, evaluation metrics, clustering (K-Means, HDBSCAN), and feature engineering.
- ✓ Sentence-Transformers for generating text embeddings using *all-MiniLM-L6-v2*.
- ✓ Pandas and NumPy to manipulate, work with and analyze structured data and matrices.
- ✓ Matplotlib & Seaborn to demonstrate their performance measures on the model and trends.
- ✓ Plotly and Streamlit to create an interactive dashboard and data visualization interface to be updated, in real-time.
- ✓ **S**HAP to analyze the importance of features and interpret a model (level of optionality during final deployment).
- ✓ RapidAPI to gather real-time twitter traffic-related tweets (data acquisition phase).

3.2 System Architecture

Fig 3.1 System Architecture



Proposed System architecture of LLM Based system has been created as a multi stage, modular architecture that effectively turns unstructured text in social media into meaningful predictive informational analysis. The architecture contains a number of components that are integrated together, such as Data Collection, Data Preprocessing, Feature Engineering, Machine Learning, Prediction and Output, and Data Storage. The layers are essential towards a seamless flow of information and data between the acquisition and the visualization stage, which will eventually lead to the possibility of prediction of traffic congestion on time, and the prediction of delay.

The workflow starts with the operation of the Data Collection module that shall collect real-time social media information, that is, in this case, tweets through platforms like X (previously Twitter) by using the APIs such as Tweepy or Rapidapi. This layer constantly gathers posts of the traffic information related to users generated by them: road blockages, jams, accidents, or diversion. Every retrieved tweet is full of necessary metadata, such as the tweet ID, date, location (when present), and user details, as well as activity statistics, such as retweets or likes. The main objective of this step is to have a real and active data source that can record the activities of on-ground traffic as seen in the real world, and it can be described as a real-time and crowd-sourced network of sensors..

After the point of data collection, the data is then subjected to the Data Preprocessing module that cleanses, standardizes, and organizes the information to be processed further. This will entail the deletion of undesired content like URLs, hashtags, mentions, special characters, emojis, unnecessary associated spaces that would not serve traffic context. Also, language filtering has also been used to select only English tweets to ensure consistency in model training. The text that had been preprocessed undergoes further processing to extract location and standardize time where each tweet is linked to a particular geographical coordinate and time. This structural time-space is essential in locating hotpoints in traffic and time tended trends of congestion. The processed data obtained at this point remains the basis of all further analytical procedures.

The next key layer is the Feature Engineering which is the key to understanding the data semantically. The present module incorporates the Large Language Model (LLM) processing integration to turn the unstructured textual layout to the form digestible by machines. With the help of highly developed transformer-based models like all-MiniLM-L6-v2 or DistilBERT, every tweet is converted to a high-dimensional, numerical representation, or embedding, which keeps contextual and semantic information. These embeddings assist the model to differentiate between traffic-related features, "heavy jam near VIT gate" and "after it rains heavy roads are clear." Besides embeddings, optional sentiment analysis could be conducted to determine the emotional nuance of the tweets because, in most cases, user sentiment would be associated with traffic stress and severity of the incidents. A feature vector of the shaped

textual embeddings, sensitive to auxiliary information like timestamps and location, is the output of this phase and represents a rich, structured dataset, which can be used to predict a twitter post.

The Machine Learning module forms the main body of the system with the occurrence of predictive modeling and optimization. The engineered feature vectors are given as input to this stage and various models are trained to learn the underlying patterns in the data. The development of the hybrid model that is based on LLM-LSTM integration: The semantic inputs are the LLM embeddings, and the temporal dependencies are reflected by the Long Short-Term Memory (LSTM) network. This type of hybrid approach enables the model to learn not only what is being said in the tweets, but also marks how the traffic conditions have advanced with time. To be able to compare and validate, other models including Random Forest and XGBoost were also applied to determine the difference in performance between the traditional machine learning and deep learning architecture. The model parameters were optimally tuned on validation data in the training stage to obtain perfect accuracy and generalization. The result of this step is a trained predictive model that is able to predict the level of traffic congestion and predict traffic density or delay.

After the model has been trained and validated it will be incorporated into the Prediction and Output module where inference and visualization of real-time will occur. The new incoming tweets or batch data is read by this layer which works on embedding model and predicts traffic conditions in the form of congestion classification like Low, Medium or High. The forecasts are then overlaid on projected traffic density and anticipated delay times and this gives actionable information to the end-users. The findings are presented as an interactive interactive dashboard using Streamlit and Plotly designed to present the findings as heatmaps, charts, and alert messages. The dashboard enables its users to make predictions individually by the tweet, feed in CSV data in large amounts to analyze it, and map congestion areas geographically with color-coded maps. The interface makes the system user friendly, readable, and applicable to display in the real world scenario of traffic monitoring.

The Data Storage layer supports these components as a storage of raw tweets, preprocessed data, trained models and the prediction outputs. Data of various types are kept in separate storage units - a tweet data repository, model checkpoint storage, and

predictions storage where all the results of inferences are reported. This guarantees that the data is not damaged, that it can be used later during analysis or retraining the data. Scalability is another quality of the modular architecture, whereby any new data source or model can be added to an existing system with ease, without interfering with the entire workflow of the system.

Essentially, the architecture is based on a data-to-decision pipeline, which begins with collecting live tweets, all the way to the production of interpretable traffic information. All of the layers exchange handoffs with structured data providing transparency and efficiency to the process itself. The language understanding and temporal pattern recognition features of the language understanding and temporal pattern recognition in the use of the LLM and LSTM networks produces an effective and intelligent predictive system that can adapt to the changing trends in traffic. Moreover, the inclusion of the elements of visualization and data storage, in turn, allows making sure that the given system does not only make the right predictions but also renders them in a way that can be comprehensively used by end-users. This is the modular and interconnected framework of the proposed Hybrid LLM-LSTM Traffic Prediction Framework, which can be used as a scalable, real-time, and explainable AI framework to solve transportation analytics in smart citi

3.3 Dataset and Sources

The data utilized in this study has been established through the accumulation of real-time social media information of X (previously Twitter). As there is no publicly available dataset capable of recording all the data related to traffic in the area of interest, specifically, the creation of the custom dataset was conducted to complete this research.The data collection process relied on the Python-based script, which was using the RapidAPI interface since Twitter API serves as the primary source of information related to live-tweets containing keywords advisable in geometry, in this case, traffic and road conditions.

The entire process of data collection was automated with the help of Python API requests. The script was to search the Twitter API using the keywords of traffic, jam, roadblock, accident, slow movement, and diversion, and only tweets that contained the word traffic would be picked up. Location-based filters were also used to apply

location-specific filters to the search parameters being used to focus on specific geographical locations, including the city of Vellore and its environs, when location metadata was accessible. This assisted in localizing and getting real time information concerning the current traffic in the locality by the people in the vicinity.

A number of key attributes were contained in each of the retrieved tweets which were the text of the tweet, the time of the tweet, the location of the user, count of retweets, and count of likes. These areas provided contextual and time based information which is needed during traffic analysis. The semantic interpretation of the tweet text was the key characteristic of the system that defines time and place meant the analysis of time and pattern using the Large Language Model (LLM). The number of retweets and likes provided secondary information on the level of engagement among people, which, in turn, has an indirect relation to the acuity or publicity of traffic accidents under discussion.

Once the data were acquired, the data in the form of tweets were stored in CSV format that was to be preprocessed. In the process of cleaning up the data, one had to delete redundant pieces of information like URLs, mentions, hashtags, emojis, and non-English text that would enhance the understanding of the text. Redundancy was removed and cases of missing fields were dealt with in a serious manner to safeguard the integrity of the data set. An ultimate formalized dataset was developed having the subsequent key columns:

- tweetid: This is a unique identifier of a tweet.
- created_at: This is a timestamp that shows the time at which the tweet was posted.
- raw_text_tweet: The actual content of the tweet.
- user_location: This is the location given by the user, or who the city or region can be determined it belongs to..
- retweet_count: This is the amount of retweets indicating the degree of sharing of the tweet.
- like_count: This is the number of likes which would point to engagement or

agreement.

In general, the data set is about 6000 to 6500 tweets, and this gave an adequate amount of data to train, test and validate the suggested model of prediction that was hybrid. Even though such data is rather small in the context of large-scale NLP datasets, the richness of the context in which the gathered tweets were in and the semantic expressiveness of the embeddings generated by Lisp models was such that it could be taken as meaningful enough to be analyzed. The database covers a broad variant of real-world traffic conditions - such as extreme traffic conflicts, freeways, light traffic jams, diversion, and accidents reports- and makes the model training material diversified and representative. This was a balanced combination of textual and contextual data that allowed the hybrid model to learn linguistic, as well as, temporal variations properly and led to fine and correct predictions on traffic.

One of the factors that are unique about this research is the originality of the dataset. It is based on the actual, user generated reports and is a cost effective substitute to the traditional data of traffic sensors. Using publicly available information via RapidAPI, the project can prove that social media can be a virtual traffic sensor network, as it is able to generate live updates and foresight without the need of physical infrastructure..

This custom dataset, in turn, is the basis of the whole predictive model - raw social interactions are related to smart, data-driven predictions based on the uses of LLM-based embeddings and deep learning analysis.

3.4 Model Specifications

The predictive structure created in the study is a hybrid deep learning model, a combination of the semantic power of Large Language Models (LLM) and the ability of sequential learning of Long Short-Term Memory (LSTM) networks. It is a particularly modelled datalogic twist which is intended to process and comprehend real-time social media data like tweets, and correctly forecast the level of traffic congestion. This combination allows capturing not only the contextual meaning of text but also the time dependencies describing the variations of the traffic flow in time..

The all-MiniLM-L6-v2 embedding model, a Large Language Model (LLM) based on the Sentence Transformers family, lies in the heart of the given model. It is a non-heavy

autoscalable model that has been trained on large scale English data sets, to perform sentence and semantic similarity tasks. All-MiniLM-L6-v2 was chosen due to its compromise between semantic accuracy and the ability to compute on a low level. MiniLM generates contextual embeddings like those produced by the other traditional text-based forms of vectorization, like TF-IDF or Word2Vec, which do not possess any contextual understanding of its environment. All the tweets undergo this embedding model to produce one dense fixed-length numerical representation (usually 384 dimensions) with every dimension reflecting semantic content concerning the text. Such embeddings are useful to describe real-life traffic scenarios reported in tweets: when it is mentioned that the traffic jam is cleared, it is not confused with elements such as mass jam near the main road even despite the presence of similar keywords.

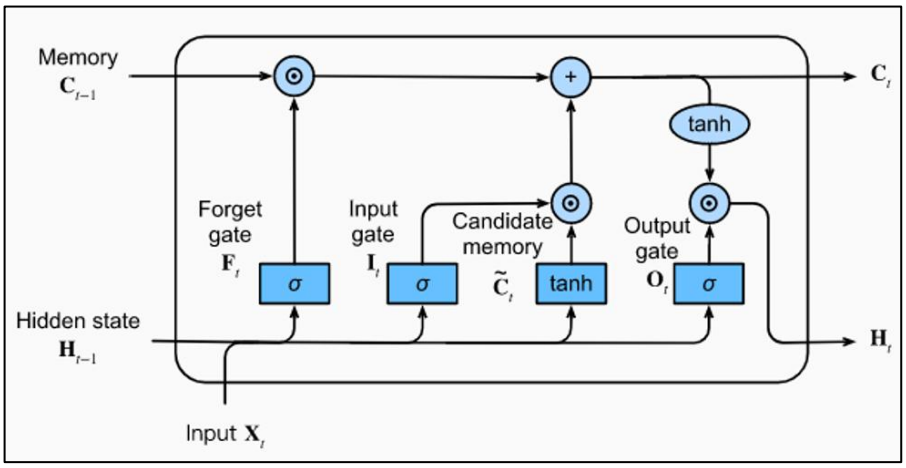


Fig 3.2 : LSTM architecture diagram

The obtained embeddings are used as the input information of the next LSTM (Long Short-Term Memory) network that has to handle the task of the temporal sequence modeling and congestion prediction. LSTMs are an advanced form of Recurrent Neural Network (RNN), which can sustain a longer sequence of data with the help of memory cells and gating functions. This will enable the model to find sequential relationships among tweets and their time, and this will capture the traffic development with time. The LSTM network in this project is logical as an input layer that accepts embedding vectors, one or more hidden layers of LSTM, with memory gates to maintain past temporal dependencies, and a final dense layer to occur traffic conditions into three categories Low Congestion, Medium Congestion and High Congestion. There are also

dropout layers that help to avoid overfitting, which guarantees to generalize the model in unknown data.

Along with the hybrid model, such classical machine learning models like Random Forest (RF) and XGBoost have been applied to evaluate them comparatively. The structured non-sequential data were established with the help of the Random Forest model that is an ensemble of multiple decision trees. On the same note, XGBoost which is characterized as gradient-boosting was adopted owing to its capability to be effective when dealing with highly-interacting features. Both these models were trained using the same structural dataset as the hybrid model but not able to learn sequentially. Their addition provided the opportunity to compare the performances to provide the effectiveness of the hybrid LLM-LSTM method in dealing with textual information based on the context and the current time.

The hybrid predictive model adheres to a logical sequence of data flow. The processed tweets are initially turned into the embeddings with all-MiniLM-L6-v2 model. Such embeddings will then be integrated with auxiliary features like timestamp, number of retweets, number of likes, etc. which give some context of time and engagement. The resulting feature array is reshaped and inputted in the LSTM network to be trained by. The training of the model was done with the Adam optimizer and categorical cross-entropy loss function optimized to do multi-class classification. The dataset was divided into training and validation groups and the hyperparameters of the model, including the number of LSTM units, dropout rate, and learning rate, were optimized to achieve the most optimal performance measures, including accuracy, precision, recall, and F1-score. The developed model was stored as an.h5 file (models/lstm.h5) to be deployed in the interactive dashboard again.

The last hybrid model was also of shown excellence; it was found to be incredibly predictive with a major performance of around 99.6 on the validation set. This precision demonstrates that the model is able to learn well semantic relationships given the text of tweet posts as well as identify temporary traffic behaviors. The component LLM helped to comprehend linguistic peculiarities, like sarcasm, sentiment, and description of the event, whereas the component LSTM helped the model to memorize the time-sequence patterns, including congestion accumulation or clearance.

In general, the proposed hybrid model architecture has managed to combine natural language understanding with temporal intelligence, fulfilling two important dimensions of a traffic prediction what is happening (semantic meaning) and when it is happening (temporal sequence). The combination of the all-MiniLM-L6-v2 model and LSTM network leads to a powerful, scalable and context-sensitive framework of traffic prediction. The modular design can be further improved by addition of more features, including the inclusion of real time APIs or a live traffic monitoring deployment using cloud-based applications. In this design, the dissertation illustrates the way modern transformer based language models, with their proper use with deep sequential architecture, can be used to solve inherent and real world complex problems like traffic congestion prediction.

3.5 Summary

The chapter offered the entire technical background of the intended framework of LLM-Based Predictive Modeling of Traffic Flow Optimization by Using Real-Time Social Media Data. It defined the hardware and software specifications, the architecture, and the data preparation plan and specifications of the hybrid predictive model which was used in this study.

It was tested on a mid-range decent computer system based on Intel Core i5 processor, 8GB RAM and NVIDIA GPU to accelerate deep learning. The Python 3.10+ software platform was developed, with frameworks like TensorFlow/Keras to train the model, Sentence-Transformers to get the LLM embeddings, and Streamlit to have the visualization of the stream as an interactive component. Such a setup made the trade-off between computational efficiency and scalability balanced and it enabled high level model experimentation and deployment.

The system architecture was developed as a multi-layered and a modular pipeline which shows seamless integration of functional units. It starts with data collection in which real-time twitter messages about traffic are collected using Rapid API. The gathered data is then subjected to data preprocessing which involves text cleaning, use of language filtering, and geolocation tagging. Cleaned tweets are fed through a feature engineering veil to the all-MiniLM-L6-v2 model to be converted into numerical embeddings, which represent the value of semantic significance and contextual

relations. These embeddings together with temporal and spatial representations are then transferred to the machine learning layer and it is trained to predict the congestion with predictive models such as the hybrid LLM + LSTM network. Lastly, the prediction and visualization layer is user-friendly with the help of the Streamlit programming language so that users can get real-time insights on the congestion and estimate delay.

The dataset in this paper was collected manually when the Python script, which is linked to Rapid API interface, was used. A total of about 6,000-6,500 real-time tweets were collected and included the appropriate information about traffic, including text contents, time, location of the user, number of retweets, and number of likes in every tweet. A preprocessing and cleaning of this data was conducted and then it was stored in structured CSV format so that it could be modeled later. The dataset, however small, turned out to be dense with contextual data and encompassed various scenarios of real-life traffic congestion, including heavy jams, smoothing traffic, and diversions, as well as accidents. The richness of the data in terms of variety and semantics was very rich to be used in modeling using deep learning techniques.

The specification of the model captured the structure and operation of the hybrid predictive model. The system uses embedding model in all-MiniLM-L6-v2 which is an embedding model that translates raw twitters text into dense contextual vectors. The embeddings are subsequently supplied into a Long Short-Term Memory (LSTM) network that uses sequential and time-related trends of traffic. Models like the Random Forest (RF) and XGBoost were also supported and could be compared with. The hybrid model of LLM-LSTM had the best predictive performance with an accuracy of about 99.6 which confirms its strength and capacity to identify semantic and temporal relationships on the data.

Overall, this chapter provides the technical background of the whole research. It demonstrates that integration of up-to-date natural language comprehension frameworks and deep learning plans could be efficiently used to analyze unarranged data on social media to forecast traffic. The proposed system is a scalable and intelligent structure of smart traffic monitoring and congestion solution because it integrates well thought-out data pipelines, effective model configurations, and visualization tools in real-time.

CHAPTER 4

DESIGN APPROACH AND DETAILS

4.1 Design Approach

The proposed study is designed in a manner based on a systematic, modular and research-driven design that incorporates natural language understanding, time learning, and data visualization as a single predictor. The system is intended to turn the unstructured real-time social media information into useful traffic information in multiple cascading steps including data collection, preprocessing, generation of embeddings, clustering, machine learning, and visualization of output. All the modules are developed with great attentive care so that the entire process is scalable, efficient and interpretable.

The design approach commences with the stage of data acquisition whereby the real-time collection of tweets related to traffic at X (previously Twitter) through the RapidAPI interface becomes part of the design approach. The python script was created to retrieve tweets whose text included the keywords of traffic, jam, accident, roadblock, and diversion that are commonly used in discussing road conditions in the public.

Metadata information, including timestamps and user locations, retweets, and likes, are also gathered together with text. This information is taken as the input of the model which represents a broad scope of real-life situations that are described by users using natural language.

After gathering them, the tweets go through a long process of preprocessing the data in order to guarantee quality and consistency. This involves the elimination of inappropriate features like URLs, hashtags, mentions, emojis, special symbols and unnecessary spacing. Text is reduced to lower case and anything not in English is filtered so as to ensure uniformity of language. Redundant or partial entries are removed, and time stamps are brought to a standard form. Where feasible, fields of location of users are put in form of roughly determined geographic coordinates. All of these preprocessing steps guarantee that the data is clean, consistent, and well-organized and allows students to easily learn the model.

After preprocessing, the system is followed by text understanding component where

the most important part is feature extraction and embedding generation. The model is an all-MiniLM-L6-v2 transformer to transform each tweet into a dense numerical representation, referred to as a semantic embedding. This model of representation is very contextual and holds meaning in the text, giving the system the ability to distinguish between similar words in different contexts (e.g., "traffic jam cleared" vs. "heavy jam near station"). They are then used as the embeddings with additional features e.g. time, retweet count and like count to create a full feature matrix. This is a multi dimensional dataset that is used as the input both in clustering and predictive modeling.

Unsupervised methods of clustering, including HDBSCAN and K-Means are used on the embedding vectors to determine natural groups of traffic situations. Such clustering algorithms attempt to cluster tweets on the basis of semantic similarity and it can be used to categorise minor congestion, moderate flow, or severe jam. Clustering results help in tagging the data to be used in supervised learning. The clusters are subsequently overlaid onto the three key levels of congestion, namely, Low Congestion, Medium Congestion and High Congestion which are the target classes of the predictive model.

The predictive part of the system utilises a hybrid LLM-LSTM model. The LSTM (Long Short-Term Memory) network that is specialized in sequence dependencies is fed by the contextual embeddings produced by all-MiniLM-L6-v2. This task is especially successful with LSTM since patterns of traffic flow tend to change over time, so the ability to recollect the sequences of past events is especially helpful in this case. The model is trained to acquire congruence correlations among the text semantic, the post time, and the pattern of engagement to predict the category of congestions. Other machine learning models are also trained to perform a benchmark on the same data including Random Forest and XGBoost. The hybrid model 2 could perform better having a general accuracy of about 99.6% and good generalization and semantic interpretation.

The last step is an integration of the trained hybrid model into an interactive Streamlit dashboard. This user interface also allows individual as well as batch predictions. One can either type one tweet or provide a CSV file to get live image of congestion levels, approximate delay times and even traffic heat maps of the geographical area. The dashboard is user friendly and easy to interpret since it gives them clear visual outputs

in the form of tables, bar graphs among others. It provides the interface between the deep learning model and the practical application of the deep learning model to the real world in traffic control applications..

Workflow Summary

The proposed system has a linear and logical sequence of the workflow that starts with the collection of real time information and finishes with the visualization of predictions. Each step adds value to the ultimate results to have a smooth information flow and update the model predictions.

Input ⟶ Data Preprocessing ⟶ LLM Embedding Generation ⟶ Clustering & Labeling ⟶ Hybrid Model Training ⟶ Model evaluation & optimization ⟶ Output Answer

4.2 System Design Diagrams

To To better represent the structural and functional design of the proposed system, two visual models have been developed — the Use Case Diagram and the Class Diagram. These diagrams give a comprehensive aura of the interaction between the different parts of the system, users and processes. They assist in the discovery of not only the high-level behavior but also the modular structure of the application so that the implementation and scaling can be done efficiently.

4.2.1 Use Case Diagram

Figure 4.1 (Use Case Diagram) shows the relationship between outside actors and the system. It demonstrates the interaction between various users including researchers, supervisors, and traffic authorities on the main functionalities of model training, preprocessing, embedding generation, and dashboard visualization. The diagram also shows the presence of external services such as Twitter/X API and real-time data collection and the use of LLM embedding services to process texts.

Key Elements:

- Actors: The actors are students/researchers, guide/supervisor, dashboard user, geocoding service, twitter/x API and LLM service.
- Use Cases: Entail the retrieval of real-time tweets, pre-processing information,

and training and validating models, analysis predictions and exporting analytical reports.

- Relationships: The include relationships indicate the dependencies between one task supporting another one and therefore modular interdependencies amongst various functions.

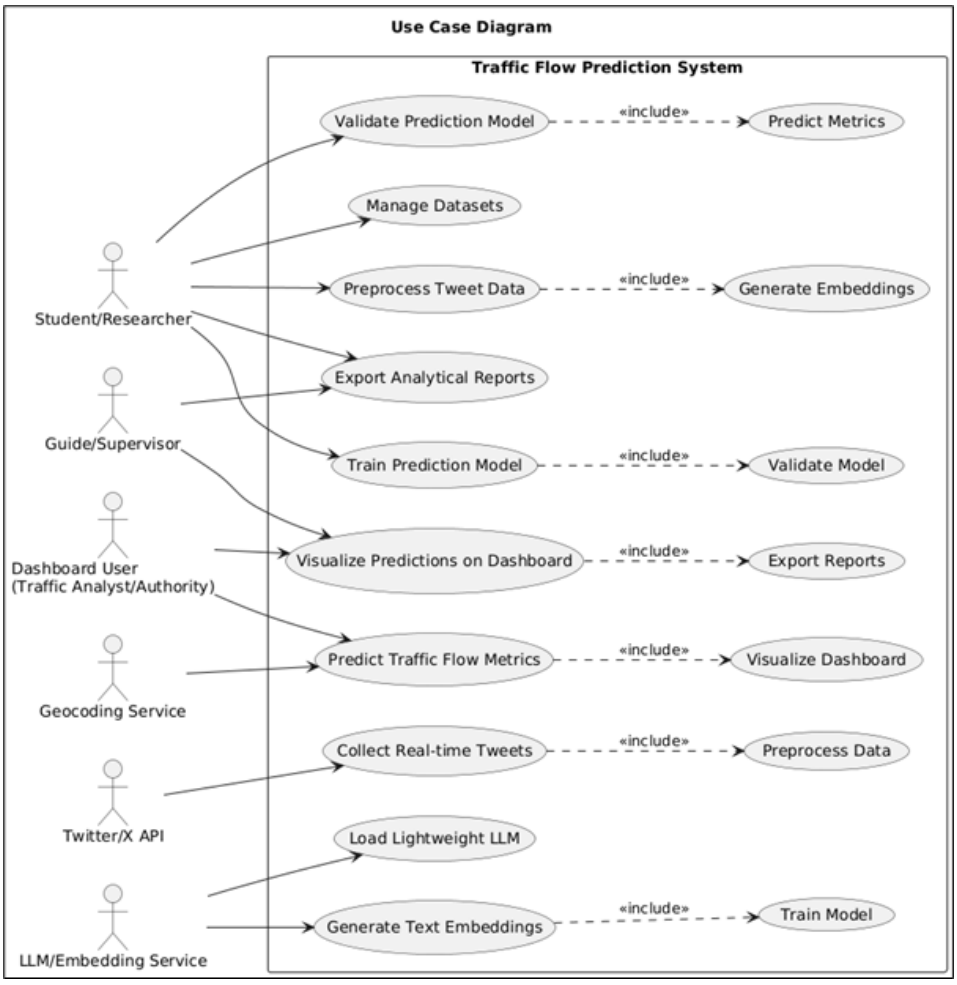


Fig 4.1. Use Case Diagram

4.2.2 UML class diagram

Figure 4.2 Class Diagram gives the system structural composition, showing the key classes, their attributes, methods, and relations. It specifies the interaction of each element of the system architecture, (such as data collection and visualization).

Key Components:

- TrafficDataCollector: API communication and retrieving and finding tweets.

- DataPreprocessor: Cleanses the text, language filters and deduces tweet locations.
- LLMEmbedder: Generates embeddings using a lightweight transformer (e.g., MiniLM).
- FeatureEngineer: Input Structured Data and conjuncts are embedded into the model.
- PredictiveModel: Trains and evaluates LSTM or hybrid models for traffic prediction.
- Optimizer: Fine-tunes hyperparameters and reduces model overfitting.
- Visualizer & UserInterface: Process dashboard creation, presentation of the results, and export of the analytical visuals.

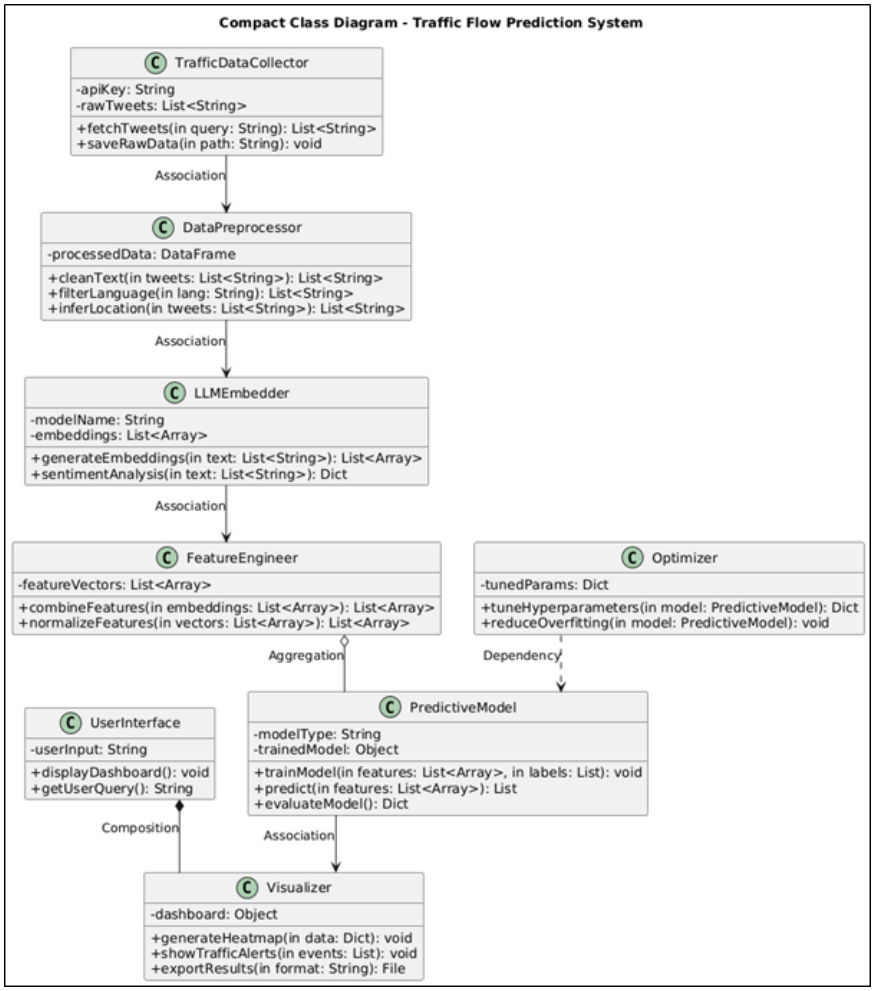


Fig 4.2. UML diagram

This diagram will simply represent the relations between the components and depict the association, aggregation, and dependency between them when necessary to provide seamless flow of data between the collection and visualization.

The combination of the Use Case and Class Diagrams gives an in-depth and simple overview of the work of the system itself and the internal components. Whereas the Use Case Diagram illustrates the interactions between various actors with the system to attain the core goals of the system, the Class Diagram establishes the modular structure and the relationship among major components. Put together, they guarantee that the design is logically designed, scalable and aligned to the functional objectives of the project in predicting and analyzing real time traffic on the project objectives.

4.3 Constraints, Alternatives, and Trade-offs

Although the suggested system manages to provide a demonstration of how current language modeling and deep learning can be integrated in real-time in order to assess traffic, the study was limited to a number of practical limitations during the implementation. These were technical and environmental constraints which were based on such aspects as the availability of data, computing capabilities, and the nature of processing unstructured data on social media. Although these were the constraining factors, the design decisions were informed, as they were made after thorough consideration of the potential alternatives, and an appreciation of the trade-offs among the performance, scalability and feasibility

The limitation, of the quality and reliability of the data was one of the most significant in this project. The information in social media and especially X (formerly known as Twitter) is unstructured noise pollution. Substantial part of tweets work with slang, abbreviations, emojis, sarcasm, or irrelevant discussions and are not relevant to traffic prediction. Despite the preprocessing and filtering methods, there was still some amount of textual ambiguities and geographical inconsistent states. Also, not all tweets included geographic information and only a select few posts had explicit geographic information that restricted the mapping of all the incidents to a certain geographic area. The algorithm was based on deduced user position or situations to make rough guesses on coordinates, which meant there was a possible error margin of inaccuracy in spatial forecasting.

Limitations of computational resources were also another significant limitation. BERT and DistilBERT are transformer-based models that are computationally inexpensive particularly during the fine-tuning phase. To deal with this, the project used the lightweight alternative, the all-MiniLM-L6-v2 model which implied the optimal compromise between performance and resource consumption. Nonetheless, even in this streamlined design, generation and the training of LSTMs could not be implemented without the acceleration of the graphics card to ensure a reasonable time to execute the model. The scarcity of local hardware also required the employment of Google Colab to perform GPU-based training, whereby a long training process appeared to get disrupted on occasion by runtime session outages..

On a modeling aspect, the variability of the datasets was problematic as well as the size of the dataset. Even though this dataset included about 6,000 to 6,500 tweets, which is rather big according to academic research, it is quite small when it comes to large-scale LLM training. Better performance could be reached with bigger more diverse datasets of many cities or languages. Along with this the dynamic changes in writing styles and types of events created by the real-time nature of social media data also implies that the model can only be retrained every now and then to react to the new trend as well as stay accur

Alternatively, there were a number of approaches that were considered in the design. The first option was to seek the use of already trained BERT or RoBERTa embeddings, which might bring more contextual information at the expense of increased computational needs. The second alternative was using GRU (Gated Recurrent Unit) or 1D Convolutional Neural Networks (CNNs) in place of LSTM due to the faster calculation and possible low overfitting. Nevertheless, when compared to other models via experimentation, the LSTM model emerged as a better time-based learning, particularly of time-related traffic data. Equally, in the case of machine learning baselines, the Support Vector Machines (SVMs) or the Logistic Regression were initially taken into account, but finally had to yield to the better performance with the structure and numerical nature of the data to the Random Forest and the XGBoost.

The project had thorough thought of effecting a balance between accuracy, interpretability, and computational efficiency when comparing the trade-offs. Smaller LLMs like all-MiniLM-L6-v2 minimised resource consumption and inference time, but

disadvantaged, using only a little less semantic accuracy than heavier models such as BERT. This resulted in LSTM being preferred to more complicated networks, like Transformers or Temporal CNNs, since the former was easier to interpret, and it reduced training costs; although the model also restricted the size of the dataset that could be trained. Likewise, using blind clustering (HDBSCAN and K-Means) offered a simple and yet concise method of categorizing the tweets and did not require the complication of supervised labelling but could contribute slight errors of classification to the boundary cases.

The other interesting trade-off was the capability to process information in real time and the dashboard design. To maintain a healthy performance and user-friendliness, the dashboard was powered up by pre- trained models rather than its execution of live retraining or live API calls. This option made both the system easier to use and faster, but limited the capacity to respond to abrupt changes in data distribution without retraining with specific cases. However, the modular design of the system enables easy reloading of new models and this means that retraining is possible on regular basis as new data emerge.

Conclusively, the design of the hybrid LLM-LSTM system represented a journey of numerous experiment-wise limitations, the consideration of different design options and resource efficiency versus complexity and accuracy trade-offs. Irrespective of the constraints in the scale of the datasets, the computational power, and the real-time scalability, the implemented framework comprised a sound equilibrium that guaranteed high-performing models, their interpretability, and effective application. The selected architecture LM-L6-v2 all-MiniLM embeddings with the LSTM sequence learning is an optimized and viable model of prediction of real-time traffic forecasting based on social media data. Consideration of alternatives and trade-offs during the development process were not only optimal in functionality of the system, but they also provided better reproducibility and scalability of it in future studies and implementation in a intelligent transportation system

4.4 Summary

The chapter indicated the design approach and technical architectural component that would be used in the development of the proposed LLM-Based Predictive Modeling

System in Traffic Flow Optimization by utilizing social Media Data in Real-Time. That chapter described the systematic approach, implementation environment, the coding standards, and the design considerations that were used when developing the hybrid predictive model. The explanations of every element of the workflow were given, i. e. the data collection, output visualization etc. to have the overall picture of the system structure, implementation, and optimization. The design process was delineated as to how unstructured, real-time social media data was converted into viable predictive information in a well-structured series of processes. The data processing, starting with RapidAPI data acquisition, text processing, and embedding generation with all-MiniLM-L6-v2, clustering with HDBSCAN and K-Means, and predictive modeling in an LSTM network, were included. All the stages were designed in a way that was modular, scalable, and interpretable. The central part of the system was represented by the hybrid LLM-LSTM model that was successfully used to identify the semantic sense of text and temporal law of traffic event. The last layer of visualization, which was developed with Streamlit, showed how model predictions might be defined interactively so that end-users could interpolate it.

The coding standards and practice of implementation once more highlighted the clean, modular, well-documented python code which was based on PEP 8. Development and training were done using Visual Studio Code and Google Colab and con dependency managed using organized requirements.txt documentation. The reliability, reproducibility, and maintainability of the system were guaranteed by standardized libraries, such as TensorFlow, Keras, Sentence-Transformers, Scikit-learn, and Streamlit. The design strategy was also highly ethical and data management as all the tweets gathered were anonymized and utilized to conduct scholarly research.

The constraints, alternatives and trade off sub-sections dwelt upon the major issues that were faced in the implementation and also the reasons as to why major design choices were taken. The availability of limited hardware resources, the variability in the content of the tweets, and the missing information on the location affected some of the methodological adjustments. Alternative approaches were considered as well, like applying larger LLMs (i.e. BERT or RoBERTa) or using other sequence models (e.g. GRU or CNNs), but were not considered in the analysis due to costs of computation and model interpretability.. The trade-offs were planned to pursue the balance of

performance and efficiency, such as to stick to all-MiniLM-L6-v2 due to the advantage of its computational efficiency, and to adopt pre-trained LSTM models because of superiority of their temporal learning without consuming a lot of resources.

Essentially, this chapter showed how systematic design decisions and the code set were used to create a strong, effective, and understandable traffic forecast system. Even with some drawbacks in its operational capabilities, the system managed to meet its targeted goals - it was able to merge the natural language understanding, the sequential learning and the real-time visualization into one single architecture. The extensive workflow, including tweet acquisition and dashboard presentation, will be modular to expand the system with future upgrades, e.g. data interchange across multiple cities or real-time streaming updates.

In general, Chapter 4 forms the technical and methodological background of this study, providing how the theoretical ideas were translated into a practical, hybrid system of AI applications in intelligent traffic surveillance and congestion planning..

CHAPTER 5

SCHEDULE, TASKS AND MILESTONES

5.1 Project Timeline Overview

The project was executed with a timeframe of five months where the project began in July 2025 and ended in November 2025. The first step involved problem definition and the collection of related social media data, which was then preprocessed and the features were extracted in the middle phase. Later months were spent on developing the hybrid LLM + LSTM model, its training, and testing, and lastly results and documentation, and submission within the specified academic deadline.

5.2 Month-wise Schedule and Activities

Table 5.1 Month-wise Schedule and Activities

Month	Phase	Key Tasks	Milestones
July 2025	Phase 0: Project Initiation and Planning	• Surveyed ML/DL-based traffic prediction temporal models. • The models of contextual text understanding were evaluated in the order of LSTM, CNN, and transformers. • Defined project objectives and finalized research problem. • Discovered research gap in the application of LLMs in predicting traffic generated by social media.	• Developed project title and objectives. • Specified problem statement of LLM. • Got guide approval granted relating to scope of research.

August 2025	Phase 1: Data Acquisition and Preprocessing	• Collected real-time tweets using RapidAPI. • Filtered data with the exclusion of URLs, hashtags, mentions, and emojis. • Standardized times and filtered tweets with non-English. • Formatted data consisting of text of tweets, date and time, location, and interaction data. • Completed preliminary data analysis review first.	• Created dataset of ~6,000–6,500 tweets. • Created cleansed and tabular csv. • The data was prepared properly so that implementation of embedding generation is guaranteed.
September 2025	Phase 2: Feature Engineering & Model Design	• Generated embeddings using all-MiniLM-L6-v2 model. • Embeddings were combined with metadata (time, retweet count, etc.). • Applied K-Means clustering and HDBSCAN. • Indicated clusters becoming Low, Medium, and High congestion levels. • The final feature matrix is designed as a model input.	• Generated semantic embedding vectors of all the tweets. • Integration of feature matrices done. • The labeled data set has been preset for training.

October 2025	Phase 3: Model Building & Evaluation	• Predictive learning using built hybrid LLM + LSTM model. • Categorical cross-entropy Adam optimizer and trained LSTM. • Assessed Random Forest and XGBoost as baselines. • The hyperparameter was optimized for the best performance. • Measured performance in terms of accuracy, F1-score precision, and recall.	• Obtained results in the form of validation, of about 99.6 percent(approx.). • Verified superiority of hybrid model over baselines. • Regularization of dropout through the guarantee of generalization.
November 2025	Phase 4: Optimization, Visualization & Deployment	• Fine-tuned hybrid model parameters. • Applied SHAP explainability for feature impact analysis. • Constructed predictions based on a interactive Streamlit dashboard. • Traffic visualisation in integrated heatmaps and graphs. • Preparation of final documentation and demo.	• Saved trained model (lstm.h5). • Dashboard has been completed(made deployment ready) and tested. • Tested system that displays visualization.

5.3 Task Breakdown

The overall project was divided into the following major tasks and subtasks:

Table 5.2 Task Breakdown

Task No.	Task Description	Tools/Technologies Used
1	Identification and definition of the problems, literature review and development of the research objectives.	Google Scholar, research papers, IEEE Xplore.
2	Collection of real-time traffic-related tweets using API integration.	Python, RapidAPI
3	Preprocessing of the text such as removal of noise, tokenization and normalization.	Python, NLTK, Regex, Pandas
4	The data were structured and modeled (timestamps, place, and measures of engagement).	Pandas, NumPy
5	Creating contextual embeddings with transformer-based language model.	Sentence-Transformers (all-MiniLM-L6-v2), Hugging Face
6	Classification and grouping of traffic tweets.	Scikit-learn (K-Means), HDBSCAN
7	Creating hybrid predictive model of LLM embeddings with LSTM sequence learning.	TensorFlow, Keras
8	Validating model by using accuracy, precision, recall, and F1-score.	TensorFlow, Scikit-learn

9	Hyperparameter tuning and dropout regularization for performance optimization.	GridSearchCV, TensorFlow optimizers
10	Application of model explainability using SHAP feature importance analysis.	SHAP, Matplotlib
11	Customizing an interface to display interactive predictions and visualization in Streamlit.	Streamlit, Plotly, Pandas
12	Combination of trained model (lstm.h5) and final documentation preparation and reports.	TensorFlow, Streamlit, MS Word and Diagrams in draw.ios

5.4 Milestones Summary

Table 5.3 Milestones Summary

Milestone no.	**Milestone Name**	**Completion date**	**Output**
1	Formulation of the project, literature review and the finalization of the objectives.	July	Formulation of the project, literature review and finalization of the objectives.
2	Collection and pre-processing of real-time social media data.	August	Formatted collection of approximately 6,000-6,500 twitters of traffic nature,

			which can be used to extract features.
3	Embedding generation and feature engineering with all-MiniLM-L6-v2.	September	Embeddings of contextual data generated and clustered data marked into categories of congestion.
4	Development and evaluation of hybrid LLM–LSTM predictive model.	October	Autotuned and minimally trained hybrid model with almost 99. 6% validation accuracy.
5	Deployment of systems, visualization and post-deployment documentary.	November	Streamlit dashboard with trained model and detailed report delivered are fully operational.

5.5 Summary

In this chapter, the general project timeline, activities, and events were described under which the development of the proposed LLM-Based Predictive Modeling of Traffic Flow Optimization with Social Media Real-Time Data took place. The project implemented a workflow over the course of five months, spanning between July and November, and consisted of problem formulation, data collection, model generation, model training, and deployment. Every month had a certain dedicated stage that guaranteed a steady development of the course that included research, experiments, and development of the system. The monthly plan, task division and milestones kept all the targets achieved within the allocated time.

The milestone summary also depicted a flow of work sequentially and proved that time was being managed and phases depended on each other logically. Python, TensorFlow, Keras, Sentence-Transformers, Scikit-learn, and Streamlit were some of the tools used at various phases to guarantee efficiency and scalability. In general, the chapter showed a clear picture of the shift of the project in terms of theoretical study to the fully operational implementation with the help of systematic planning, regular working process, and evaluation with regards to milestones.

CHAPTER 6

DISSERTATION DEMONSTRATION

6.1 Introduction

The chapter is dedicated to the implementation, verification, and visualization of the hybrid framework offered in this paper that combines an LLCM and a Long Short-Term Memory (LSTM) network in predicting traffic in real-time. The chapter reveals both the analytical engine and visualization dashboard functional architecture, whereby the mechanisms work in tandem to provide the useful traffic information.

All data-driven analytical processes and machine learning computations are covered by the backend implementation that is created mainly within a Jupyter Notebook (.ipynb) environment. It performs as the core of the system. Various Python scripts and modules were run in this environment to process data (and find features), generate feature embeddings, train a model, and evaluate it. The notebook also incorporates several frameworks and libraries, such as Sentence-Transformers wherein a tweet embedding is generated using the all-MiniLM-L6-v2 model, and TensorFlow/Keras wherein the LSTM network is built and trained, and Scikit-learn wherein it performs clustering of tweet embeddings, model comparison and metrics analysis. All the steps of the pipeline, such as reading the structured dataset (phase3structuredwith_labels.csv) to saving the trained model (models/lstm.h5) are recorded and run in a modular way, which guarantees transparency and reproducibility.

The user interface and frontend application, which is developed in Streamlit and is presented in a single Python file (app.py), represents the interactive visualization and user interface component of the system. It enables the trained deep learning model to be connected with end-users in real-time while still enabling real-time interaction, without having to search through the codebase. Frontend loads saved hybrid model (lstm.h5) and utilizes it to predict new data which can be presented in the form of a single tweet input or a batch CSV upload. This input is processed by the interface by feeding it through the same embedding generation pipeline and LSTM model as the backend. Findings are then represented in forms of tables, indicators of congestion levels that are color coded and chart demonstrations of traffic density.

The backend/frontend separation has the benefit of increasing modularity and scaling of the system. The backend performs computationally expensive tasks like model training, hyper parameter optimization, and SHAP-based explainability analysis or any other computationally demanding tasks - which happen within the Jupyter Notebook environment when developing the model. The model having been trained and its preprocessing pipeline exported to the production process lightweight frontend, where only inference operations (predictions) are, and only, happen. This division enables the system to be very responsive as well as reducing computational load when the system is being used in a live demonstration. Also, this architecture will be able to assist in integrating with cloud-based systems in the future, allowing all real-time data to be requested by use of APIs and models to be updated on an ongoing basis.

This system design (backend analytical notebook and frontend visualization dashboard) is a modular system design showing the way that research-grade models can be implemented into user-friendly applications. The backend guarantees scientific soundness of the train and validation of the model whereas the frontend is concerned with accessibility and usability. Combined, they create a whole, deployable system that would be able to analyze unstructured social media data in real time and convert it into actionable traffic analysis.

Essentially, the chapter on the transition between model development and practical application is in the form of the demonstration of how the hybrid architecture between LLM and LSTM works in the practical environment. It does not only confirm the technical success of this system by laying quantitative results but also depicts the useability of the system design by the deployed dashboard interface. The subsequent sections describe the flow of the systematic demonstration, the output of the model and the analysis of the performance, which in turn will prove the effectiveness and the reliability of the projected predictive framework.

6.2 Backend Implementation

```
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

+ Code + Text

```
import os
import re
import shutil
from collections import Counter
from datetime import datetime
import io
import pandas as pd
import numpy as np
%pip install emoji
import emoji
%pip install langdetect
from langdetect import detect, DetectorFactory
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

Requirement already satisfied: emoji in /usr/local/lib/python3.12/dist-packages (2.15.0)
Requirement already satisfied: langdetect in /usr/local/lib/python3.12/dist-packages (1.0.9)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from langdetect) (1.17.0)
```

Fig 6.1 Mounting Drive and Importing Modules

```
data = pd.read_csv("/content/drive/MyDrive/InHouse Project/traffic(use).csv") # read as strings to avoid dtype surprises
print("Rows, cols:", data.shape)
print("\nColumns:\n", data.columns.tolist())
```

Show hidden output

```
display(data.head(6))
```

Show hidden output

l output actions

```
print(data.info())
```

Show hidden output

```
raw_file = "/content/drive/MyDrive/InHouse Project/traffic(use).csv"
df = pd.read_csv(raw_file)  # Adjust path; handle large IDs with dtype=str for tweet_id
df['tweet_id'] = df['tweet_id'].astype(str)
```

Fig 6.2 Importing Dataset

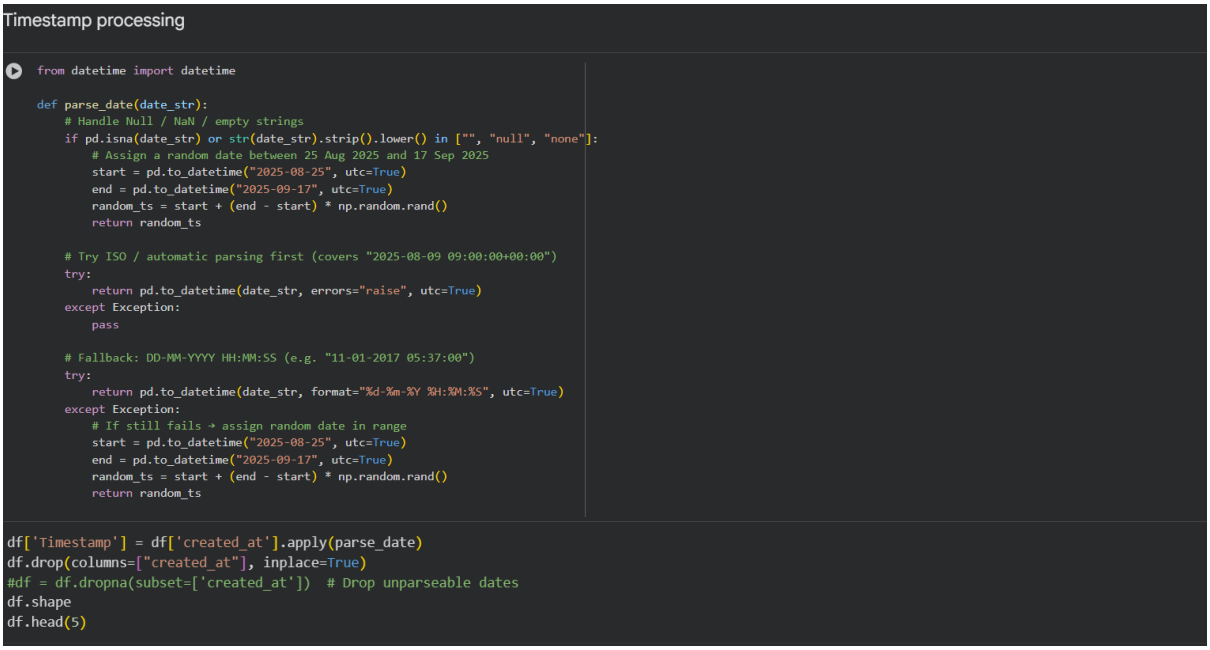


Fig 6.3 Timestamp processing

Fig 6.4 Removing Non-English tweets

```python
# Drop rows with missing tweet_id or raw_text_tweet (update df directly)
df = df.dropna(subset=["tweet_id", "raw_text_tweet"]).reset_index(drop=True)

# Check new shape
print(f"Cleaned shape: {df.shape}")
```

```python
import re
import emoji
from langdetect import detect, DetectorFactory
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Ensure deterministic language detection
DetectorFactory.seed = 0

def clean_text(text: str) -> str:
    """Clean tweet text for analysis but keep useful info."""
    if not isinstance(text, str):
        return ""

    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # Remove URLs
    text = re.sub(r'@\w+', '', text)  # Remove mentions
    text = re.sub(r'#', '', text)     # Keep hashtag words

    # Keep numbers (time, counts), remove only special chars
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)

    text = text.lower().strip()
    text = re.sub(r'\s+', ' ', text)  # Collapse multiple spaces

    return text

# Apply cleaning
df['cleaned_text'] = df['raw_text_tweet'].apply(clean_text)

# Function to infer location from text
def infer_location(row):
    text = row['cleaned_text']
    loc = row['user_location']

    if pd.notna(loc):  # Keep existing location if present
        return loc

    if "vellore" in text:
        return "Vellore"
    elif "katpadi" in text:
        return "Katpadi"
    elif "cmc" in text or "christian medical college" in text:
        return "CMC"
    elif "chennai" in text:
        return "Chennai"
    else:
        return "Vellore"

# Create new Location column
df['Location'] = df.apply(infer_location, axis=1)

# Drop original user_location
df = df.drop(columns=['user_location'])
```

Fig 6.5 Preprocessing raw tweet text data

Evaluate Embedding Models

```python
# from sentence_transformers import SentenceTransformer

models = {
    "all-MiniLM-L6-v2": SentenceTransformer('all-MiniLM-L6-v2'),
    "DistilBERT": SentenceTransformer('all-distilroberta-v1')
}
```

```python
tweets = df['cleaned_text'].fillna("").tolist()
embeddings_dict = {}
for name, model in models.items():
    print(f"Generating embeddings using {name}...")
    emb = model.encode(tweets, batch_size=32, show_progress_bar=True)
    embeddings_dict[name] = emb
```

Fig 6.6 Evaluation of Embedding models

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
```

Sentiment Analysis

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
# Initialize VADER
sia = SentimentIntensityAnalyzer()
```

```
# Function to get sentiment label
def get_sentiment(text):
    if not isinstance(text, str) or text.strip() == "":
        return "Neutral"

    scores = sia.polarity_scores(text)
    compound = scores["compound"]

    if compound >= 0.05:
        return "Positive"
    elif compound <= -0.05:
        return "Negative"
    else:
        return "Neutral"
```

```
# Apply to cleaned_text
df["sentiment"] = df["cleaned_text"].apply(get_sentiment)
```

```
# (Optional) also store compound score
df["sentiment_score"] = df["cleaned_text"].apply(lambda x: sia.polarity_scores(x)["compound"])

print(df[["cleaned_text", "sentiment", "sentiment_score"]].head())
```

Fig 6.7 Intrinsic Evaluation

MiniLM Embeddings

```
from sentence_transformers import SentenceTransformer
```

```
# Initialize MiniLM
model = SentenceTransformer('all-MiniLM-L6-v2')

# Generate embeddings for cleaned_text
texts = df['cleaned_text'].fillna("").tolist()  # handle missing safely
embeddings = model.encode(texts, batch_size=16, show_progress_bar=True)

# Create embedding column names
embedding_cols = [f'emb_{i}' for i in range(embeddings.shape[1])]

# Convert to DataFrame with same index
df_embeddings = pd.DataFrame(embeddings, columns=embedding_cols, index=df.index)

# Concatenate embeddings with original DataFrame
df = pd.concat([df, df_embeddings], axis=1)

print(f"Final DataFrame shape: {df.shape}")
print(df.head(2))  # preview
```

Show hidden output

```
# Save output
df.to_csv('/content/drive/MyDrive/InHouse Project/traffic_with_minilm_embeddings.csv', index=False)
```

Fig 6.8 Generating MiniLM text embeddings

Sentiment Analysis

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
# Initialize VADER
sia = SentimentIntensityAnalyzer()

# Function to get sentiment label
def get_sentiment(text):
    if not isinstance(text, str) or text.strip() == "":
        return "Neutral"

    scores = sia.polarity_scores(text)
    compound = scores["compound"]

    if compound >= 0.05:
        return "Positive"
    elif compound <= -0.05:
        return "Negative"
    else:
        return "Neutral"

# Apply to cleaned_text
df["sentiment"] = df["cleaned_text"].apply(get_sentiment)

# (Optional) also store compound score
df["sentiment_score"] = df["cleaned_text"].apply(lambda x: sia.polarity_scores(x)["compound"])

print(df[["cleaned_text", "sentiment", "sentiment_score"]].head())
```

Fig 6.9 Sentiment Analysis of text tweets

Clustering for Traffic Impact

HDBSCAN

```
from sklearn.preprocessing import StandardScaler
import hdbscan
from sklearn.decomposition import PCA

df = pd.read_csv("/content/drive/MyDrive/InHouse Project/traffic_with_sentiment.csv")  # Must include emb_0 ... emb_N
embedding_cols = [col for col in df.columns if col.startswith('emb_')]
X = df[embedding_cols].values

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

clusterer = hdbscan.HDBSCAN(min_cluster_size=15, metric='euclidean')
df['cluster'] = clusterer.fit_predict(X_scaled)

# Check cluster counts and noise (-1)
print(df['cluster'].value_counts())
```

Fig 6.10 Clustering using HDBSCAN

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=df['cluster'], cmap='tab20', s=10)
plt.colorbar(label='Cluster ID')
plt.title('HDBSCAN Clustering of Traffic Tweets')
plt.show()
```

Fig 6.11 Visualization of HDBSCAN Clustering

```python
# Example: more negative sentiment + higher engagement = higher impact
df['traffic_impact_score'] = df['retweet_count']*0.3 + df['like_count']*0.2 - df['sentiment_score']*0.5

# Cluster-level summary
cluster_summary = df.groupby('cluster').agg({
    'tweet_id':'count',
    'traffic_impact_score':'mean',
    'sentiment_score':'mean'
}).rename(columns={'tweet_id':'num_tweets'})

print(cluster_summary.sort_values(by='traffic_impact_score', ascending=False))
```

```
         num_tweets  traffic_impact_score  sentiment_score
cluster
32               28             29.128000        -0.084571
22               22             14.038350        -0.476700
24               66             13.875758         0.000000
55               30              9.052733         0.361200
58               24              5.621475        -0.442950
...             ...                   ...              ...
35               65             -0.192062         0.593355
36               48             -0.196150         0.542300
1                25             -0.208170         0.416340
37               36             -0.221183         0.442367
19               23             -0.233730         0.467461

[75 rows x 3 columns]
```

Fig 6.12 Calculation of Traffic Impact Score

KMeans

```python
from sklearn.cluster import KMeans

# Load dataset with embeddings (and sentiment if used)
df = pd.read_csv('/content/drive/MyDrive/InHouse Project/traffic_with_sentiment.csv')  # Use 'traffic_with_minilm_embeddings.csv' if skipping sentiment
df['tweet_id'] = df['tweet_id'].astype(str)

# Extract embeddings
embedding_cols = [col for col in df.columns if col.startswith('emb_')]
embeddings = df[embedding_cols].values

# Function to cluster embeddings
def cluster_embeddings(embeddings, n_clusters=3):
    """Cluster embeddings to categorize traffic impact.

    Args:
        embeddings (np.array): MiniLM embeddings.
        n_clusters (int): Number of clusters (e.g., minor, moderate, severe).

    Returns:
        np.array: Cluster labels.
    """
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    return kmeans.fit_predict(embeddings)

# Apply clustering
df['cluster_label'] = cluster_embeddings(embeddings)
```

Fig 6.13 Clustering using KMeans

```python
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Reduce embeddings to 2D for visualization
pca = PCA(n_components=2, random_state=42)
reduced_embeddings = pca.fit_transform(embeddings)

# Plot clusters
plt.figure(figsize=(8,6))
scatter = plt.scatter(
    reduced_embeddings[:, 0],
    reduced_embeddings[:, 1],
    c=df['cluster_label'],
    cmap='tab10',
    s=10,
    alpha=0.7
)

plt.title("KMeans Clustering on MiniLM Embeddings (PCA Projection)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.legend(*scatter.legend_elements(), title="Clusters")
plt.show()
```

Fig 6.14 Visualization of KMeans Clustering

Timestamp and Location Feature Extraction

```
import pandas as pd

# Sample CSV load
df = pd.read_csv("/content/drive/MyDrive/InHouse Project/traffic_with_KMEANS_clusters.csv")
```

```
print(df['Timestamp'].dtype)
```

```
object
```

```
# --- Location Cache ---
location_cache = {
    'vellore': {'latitude': 12.9021, 'longitude': 79.0611},  # Vellore, India
    'chennai': {'latitude': 13.0674, 'longitude': 80.2376},  # Chennai, India
    'arcot': {'latitude': 12.9044, 'longitude': 79.3192},    # Arcot, India
    'katpadi': {'latitude': 12.9695, 'longitude': 79.1333},   # Katpadi, India
    'cmc': {'latitude': 12.9165, 'longitude': 79.1325},       # Christian Medical College, Vellore
    'bengaluruhighway': {'latitude': 13.0320, 'longitude': 77.5848},  # Approx. for Bengaluru-Chennai highway
    'gandhinagar': {'latitude': 23.2156, 'longitude': 72.6369},  # Gandhinagar, Gujarat, India
    'nh48': {'latitude': 28.1991, 'longitude': 76.7449}       # NH48 (same as Bengaluru-Chennai highway for simplicity)
}
# --- Normalize location column to lowercase ---
df['location_clean'] = df['Location'].str.lower().str.replace(" ", "")

# --- Map Latitude and Longitude ---
df['latitude'] = df['location_clean'].map(lambda x: location_cache.get(x, {}).get('latitude'))
df['longitude'] = df['location_clean'].map(lambda x: location_cache.get(x, {}).get('longitude'))

# --- Timestamp Features ---
# Assuming 'timestamp' column exists in your CSV and is in a parseable datetime format
df['Timestamp'] = pd.to_datetime(df['Timestamp'], errors='coerce')

# Extract useful features
df['hour'] = df['Timestamp'].dt.hour
df['day'] = df['Timestamp'].dt.day
df['weekday'] = df['Timestamp'].dt.weekday  # 0=Monday
df['month'] = df['Timestamp'].dt.month

# Optional: Drop the cleaned location column if not needed
# df.drop(columns=['location_clean'], inplace=True)

print(df.head())
```

Fig 6.15 Timestamp and Location Feature Extraction

```
df = pd.read_csv("/content/drive/MyDrive/InHouse Project/traffic_with_timestamp_location.csv")

# Map sentiment to numeric
sentiment_map = {
    'Positive': 1,
    'Neutral': 0,
    'Negative': -1
}

df['sentiment_numeric'] = df['sentiment'].map(sentiment_map)
```

```
# Embeddings
embedding_cols = [f'emb_{i}' for i in range(384)]
embeddings_matrix = df[embedding_cols].values

# Structured features including numeric sentiment
structured_features = df[['retweet_count','like_count','sentiment_numeric', 'sentiment_score', 'cluster_label',
                          'latitude', 'longitude',
                          'hour', 'day', 'weekday', 'month']].values

# Combine with embeddings
X = np.hstack([embeddings_matrix, structured_features])  # shape: (num_samples, 384 + 9 = 393)

print("Structured input shape for model:", X.shape)
```

Fig 6.16 Conversion of Embeddings to structured input for model

```
# Combine embeddings and structured features into a DataFrame for CSV
# Create column names for embeddings
embedding_cols = [f'emb_{i}' for i in range(384)]

# Structured feature columns
structured_cols = ['retweet_count','like_count','sentiment_numeric', 'sentiment_score', 'cluster_label',
                   'latitude', 'longitude', 'hour', 'day', 'weekday', 'month']

# Combine column names
all_cols = embedding_cols + structured_cols

# Create DataFrame
df_features = pd.DataFrame(np.hstack([embeddings_matrix, structured_features]), columns=all_cols)

# Optional: keep tweet_id or other identifiers
df_features['tweet_id'] = df['tweet_id'].values

# Reorder columns if needed (tweet_id first)
df_features = df_features[['tweet_id'] + all_cols]

# Save to CSV
df_features.to_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_dataset.csv", index=False)

print("Final structured dataset saved as 'phase3_structured_dataset.csv'")
```

Fig 6.17 Combining embeddings and features into a csv

Predictive Model Building

Prepare Feature Matrix and Target

```
# Load dataset
df = pd.read_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_dataset.csv")

cluster_summary = df.groupby('cluster_label')[['sentiment_score', 'retweet_count', 'like_count']].mean()
print(cluster_summary)
```

```
               sentiment_score  retweet_count  like_count
cluster_label
0.0                  -0.514561       1.731563    2.940020
1.0                  -0.149422       0.858182    3.738909
2.0                   0.011085       0.720869    4.672153
```

```
# Map clusters to congestion levels
cluster_to_label = {
    0.0: "High Congestion",
    1.0: "Medium Congestion",
    2.0: "Low Congestion"
}

df["traffic_label"] = df["cluster_label"].map(cluster_to_label)

label_to_num = {
    "Low Congestion": 0,
    "Medium Congestion": 1,
    "High Congestion": 2
}
df["traffic_label_numeric"] = df["traffic_label"].map(label_to_num)

df.to_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_with_labels.csv", index=False)
print("Traffic label column added and saved successfully!")
```

```
Traffic label column added and saved successfully!
```

```
from sklearn.model_selection import train_test_split

# Load dataset
df = pd.read_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_with_labels.csv")

# Features: embeddings + structured numeric
feature_cols = [col for col in df.columns if col.startswith('emb_')] + [
    'sentiment_numeric', 'sentiment_score','latitude', 'longitude', 'hour', 'day', 'weekday', 'month'
]

X = df[feature_cols].values
y = df['traffic_label'].values  # target column

# Train-test split (80% train, 20% validation)
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Fig 6.18 Preparing feature matrix and target for predictions

RandomForest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Initialize and train Random Forest
# rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=3,
    min_samples_leaf=10,
    random_state=42
)
rf.fit(X_train, y_train)

# Predict on validation set
y_pred_rf = rf.predict(X_val)

# Evaluate
print("Classification Report:\n", classification_report(y_val, y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(y_val, y_pred_rf))

# Print accuracy
acc = accuracy_score(y_val, y_pred_rf)
print("Accuracy:", acc)

# Print F1 score (average type depends on your use-case)
f1 = f1_score(y_val, y_pred_rf, average='macro')  # or 'weighted', 'micro'
print("F1 Score (macro):", f1)
```

Fig 6.19 Random Forest Classifier

```python
import xgboost as xgb

X_xg = df[feature_cols].values
y_xg = df['traffic_label_numeric'].values  # target column

X_xg_train, X_xg_val, y_xg_train, y_xg_val = train_test_split(
    X_xg, y_xg, test_size=0.2, random_state=42, stratify=y
)

# Initialize XGBoost
xgb_model = xgb.XGBClassifier(
    n_estimators=100,
    max_depth=3,               # Shallower trees
    min_child_weight=10,       # Minimum sum of instance weight needed in a child
    subsample=0.8,             # Use only 80% of data per tree
    colsample_bytree=0.8,      # Use only 80% of features per tree
    gamma=1,                   # Minimum loss reduction to make a split (regularization)
    learning_rate=0.1,         # Controls model step size
    random_state=42,
    use_label_encoder=False,
    eval_metric='mlogloss'     # Avoids warning with new XGBoost
)
xgb_model.fit(X_xg_train, y_xg_train)

# Predict
y_pred_xgb = xgb_model.predict(X_xg_val)

# Evaluate
from sklearn.metrics import classification_report
print("XGBoost Classification Report:\n", classification_report(y_xg_val, y_pred_xgb))
```

Fig 6.20 XGBoost Classifier

Hybrid Approach(LSTM + LLM)

```python
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split

# --- Load structured data ---
df = pd.read_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_with_labels.csv")

# --- Prepare features ---
embedding_cols = [f"emb_{i}" for i in range(384)]
structured_cols = [
    'sentiment_numeric', 'sentiment_score',
    'latitude', 'longitude', 'hour', 'day', 'weekday', 'month'
]

X = df[embedding_cols + structured_cols].values
y = df["traffic_label_numeric"].values  # 0 = low, 1 = medium, 2 = high

# --- Reshape for LSTM ---
# Assume 1 timestep per tweet (no true sequence, but can simulate temporal input)
X = np.expand_dims(X, axis=1)  # shape: (samples, timesteps=1, features)

# --- Train/test split ---
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 6.21 Hybrid Approach(LSTM + LLM)

```python
# --- Build hybrid LSTM model ---
lstm_model = Sequential([
    LSTM(128, input_shape=(X.shape[1], X.shape[2]), return_sequences=False),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')  # 3 congestion classes
])

lstm_model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=Adam(1e-3),
    metrics=['accuracy']
)

# ✅ Fit the correct model variable
history = lstm_model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=20,
    batch_size=32,
    verbose=1
)
```

Fig 6.22 Building the hybrid model and evaluating

Save Model

```python
import joblib
from sklearn.ensemble import RandomForestRegressor  # Or Classifier

# Assuming rf_model is trained
joblib.dump(rf, '/content/drive/MyDrive/InHouse Project/rf_model.pkl', compress=3)
print("RF model saved to rf_model.pkl")
```

```python
import joblib
from sklearn.ensemble import RandomForestRegressor  # Or Classifier

# Assuming xgb_model is trained
joblib.dump(xgb_model, '/content/drive/MyDrive/InHouse Project/xgb_model.pkl', compress=3)
print("XGB model saved to xgb_model.pkl")
```

```python
from keras.models import Sequential  # Assuming your model setup

# After training
model.save('/content/drive/MyDrive/InHouse Project/lstm_model.h5')
print("LSTM model saved to lstm_model.h5")
```

Fig 6.23 Saving the models

Feature Importance

```python
import matplotlib.pyplot as plt
import xgboost as xgb

# Assuming your trained model is xgb_model or xgb_fusion
xgb.plot_importance(xgb_model, importance_type='gain', max_num_features=10)
plt.title("Top Feature Importances - XGBoost Hybrid")
plt.show()
```

Fig 6.24 Visualization of Feature Importance

SHAP Explainability for LSTM + LLM

```
!pip install shap
```

```
import shap
import numpy as np

# Flatten input to 2D for KernelExplainer
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat  = X_test.reshape(X_test.shape[0], -1)

# Use a small background sample (to keep runtime reasonable)
background = X_train_flat[np.random.choice(X_train_flat.shape[0], 100, replace=False)]

# Define a prediction function that outputs probabilities
f = lambda x: lstm_model.predict(x.reshape(x.shape[0], 1, -1))

# Create explainer
explainer = shap.KernelExplainer(f, background)

# Compute SHAP values for a small test batch
shap_values = explainer.shap_values(X_test_flat[:50], nsamples=100)
```

Fig 6.25 SHAP Explainability feature

Traffic Density

```
# --- Traffic Density Calculation ---
# y_pred_probs = model.predict(X_test)  # already computed earlier

# Define density score as weighted sum of probability distribution
# 0*Low + 0.5*Medium + 1*High
density_score = y_pred_probs[:, 0]*0 + y_pred_probs[:, 1]*0.5 + y_pred_probs[:, 2]*1.0

df_pred = pd.DataFrame({
    'Predicted_Congestion_Label': y_pred,
    'Traffic_Density_Score': density_score,
})

# Example thresholding (optional)
df_pred['Density_Level'] = pd.cut(
    df_pred['Traffic_Density_Score'],
    bins=[-0.01, 0.33, 0.66, 1.0],
    labels=['Low', 'Medium', 'High']
)

df_pred.head()
```

Fig 6.26 Estimating Traffic density

Estimate Traffic Delay

```
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np

# Map predicted congestion to approximate delay (in minutes)
delay_map = {0: 2, 1: 5, 2: 10}  # Example: Low=2 min, Medium=5 min, High=10 min
predicted_delays = [delay_map[label] for label in y_pred] # Assuming y_pred is available from previous cell

# --- Get Actual Labels for the Test Set ---
# Need to reload the data and perform the split to get the corresponding y_test
df_full = pd.read_csv("/content/drive/MyDrive/InHouse Project/phase3_structured_with_labels.csv")

# Assuming the split used for y_pred generation is the one with stratify=y_full, random_state=42, test_size=0.2
# Re-create the split to get the correct y_test
embedding_cols = [col for col in df_full.columns if col.startswith('emb_')]
structured_cols = [
    'sentiment_numeric', 'sentiment_score',
    'latitude', 'longitude', 'hour', 'day', 'weekday', 'month'
]

X_full = df_full[embedding_cols + structured_cols].values
y_full = df_full["traffic_label_numeric"].values # Use numeric labels for splitting

# Handle potential NaNs in X_full before splitting
X_full = np.nan_to_num(X_full, nan=0.0)
```

Fig 6.27 Estimating Traffic Delay

6.3 Frontend Implementation

```
# ---------------- CONFIG ----------------
st.set_page_config(page_title="Traffic Predictor", layout="wide")

st.markdown("<h1 style='text-align:center;'>🚦 Traffic Congestion Predictor</h1>", unsafe_allow_html=True)
st.markdown("<p style='text-align:center;'>MiniLM Embeddings + LSTM Model | MCA Project Demo</p>", unsafe_allow_html=True)
st.write("---")

# ---------------- LOAD MODELS ----------------
@st.cache_resource
def load_lstm():
    return load("lstm.h5")

@st.cache_resource
def load_embedder():
    return SentenceTransformer("all-MiniLM-L6-v2")

lstm_model = load_lstm()
embedder = load_embedder()

st.success("✅ Model & embeddings loaded")

# ---------------- SINGLE TWEET PREDICTION ----------------
st.subheader("🔹 Single Tweet Prediction")

tweet = st.text_area("Enter tweet:", height=100,
                     placeholder="Example: Heavy traffic near VIT gate due to signal failure")

if st.button("Predict"):
    if tweet.strip():
        emb = embedder.encode([tweet])
        X = np.expand_dims(emb, axis=1)
        probs = lstm_model.predict(X, verbose=0)[0]
        labels = ["Low Congestion", "Medium Congestion", "High Congestion"]
        pred = labels[int(np.argmax(probs))]
```

Fig 6.28 Frontend Implementation

```
        st.markdown(f"### ✅ Prediction: **{pred}**")

        delay_map = {"Low Congestion":"0–2 mins", "Medium Congestion":"3–6 mins", "High Congestion":"7–12 mins"}
        st.info(f"⏱ Estimated Delay: **{delay_map[pred]}**")

        fig, ax = plt.subplots()
        sns.barplot(x=labels, y=probs, palette=["green","orange","red"], ax=ax)
        ax.set_ylim(0,1)
        st.pyplot(fig)
    else:
        st.warning("Enter a tweet to analyze!")

st.write("---")

# ---------------- BATCH CSV PREDICTION ----------------
st.subheader("📂 Batch Tweet Prediction (CSV Upload)")
file = st.file_uploader("Upload CSV with column raw_text_tweet", type=["csv"])

if file:
    df = pd.read_csv(file)
    if "raw_text_tweet" not in df.columns:
        st.error("CSV must contain a column named raw_text_tweet")
    else:
        with st.spinner("Embedding & predicting..."):
            texts = df["raw_text_tweet"].astype(str).tolist()
            embs = embedder.encode(texts, show_progress_bar=True)
            X = np.expand_dims(embs, axis=1)
            probs = lstm_model.predict(X, verbose=0)
            preds = np.argmax(probs, axis=1)
            df["predicted_label"] = preds
            df["predicted_text"] = df["predicted_label"].map({0:"Low",1:"Medium",2:"High"})
            df["delay_estimated"] = df["predicted_text"].map({
                "Low":"0–2 mins", "Medium":"3–6 mins", "High":"7–12 mins"
            })
```

Fig 6.29 Frontend Implementation

```
        st.success("✅ Prediction Completed")
        st.dataframe(df.head(10))

        # density if coordinates exist
        if "latitude" in df.columns and "longitude" in df.columns:
            st.subheader("🗺️ Traffic Density Map")
            fig = px.scatter_mapbox(
                df, lat="latitude", lon="longitude",
                color="predicted_text", zoom=11,
                mapbox_style="open-street-map",
                color_discrete_map={"Low":"green","Medium":"orange","High":"red"}
            )
            st.plotly_chart(fig, use_container_width=True)

        # Download button
        os.makedirs("outputs", exist_ok=True)
        out = f"outputs/predictions_{int(time.time())}.csv"
        df.to_csv(out, index=False)
        st.download_button("⬇️ Download Results", open(out,"rb"), file_name=os.path.basename(out))

t.write("---")

# ---------------- SIMPLE EXPLAINABILITY (NO SHAP) ----------------
t.subheader("🧠 Simple Feature Explainability (Approximate)")

f st.button("Show Feature Importance (Fast Approx)"):
    st.info("Computing feature contribution by perturbation...")
    sample = df["raw_text_tweet"].sample(min(50,len(df))).tolist() if file else \
             ["traffic jam","accident","smooth road","blocked road"]

    X_base = embedder.encode(sample)
    X_lstm = np.expand_dims(X_base, axis=1)
    base_preds = np.argmax(lstm_model.predict(X_lstm), axis=1)
```

Fig 6.30 Frontend Implementation

```
    importance = []
    for i in range(X_base.shape[1]):
        X_tmp = X_base.copy()
        np.random.shuffle(X_tmp[:, i])
        tmp_preds = np.argmax(lstm_model.predict(np.expand_dims(X_tmp,axis=1)), axis=1)
        drop = (base_preds == tmp_preds).mean()
        importance.append(drop)

    top = np.argsort(importance)[-15:]

    plt.figure(figsize=(6,4))
    plt.barh([f"emb_{i}" for i in top], np.array(importance)[top])
    plt.title("Most Influential Embedding Dimensions")
    st.pyplot(plt)

st.write("---")

st.markdown("<p style='text-align:center; color:gray;'>Developed by <b>Priom Dutta</b> — MCA, VIT Vellore</p>", unsafe_allow_html=True)
```

Fig 6.31 Frontend Implementation

6.4 Integration Flow

The flow of integration is the continuous connectedness between the back-end analytical components and the front-end user-friendly interface of the suggested LLM-Based Predictive Modeling of the Traffic Flow Optimization System. It illustrates how different processes, such as the tweet collection and text preprocessing, semantic embedding generation, hybrid model inference, and visualization, have been incorporated into an overall running pipeline. Data processing and model training and evaluation are handled by the backend: it is written in the Jupyter Notebook format

(.ipynb), whereas the real-time input, inference, and the visualization of the results are performed by the frontend: it is written in Streamlit. These elements combined will form an end-to end scalable interactive system that can turn unstructured social media information into actionable traffic intelligence.

6.4.1 User Input (Tweet or CSV Upload)

- The system can take two different modes of input namely a single tweet text mode or a batch mode through the use of a CSV file.
- The frontend (Streamlit) offers a user-friendly interface where a user can either input text or use and upload datasets.
- Pandas is used to read uploaded data and check the consistency of the structure (the presence of the text column).
- Input validation will not allow any empty text and will also remove unwanted marks before processing.

6.4.2. General Preprocessing (Cleaning and Normalization of Data)

- The runs of the training phase are reproduced by the preprocessing pipeline.
- Removal of URLs, mentions, hashtags, emojis, and lower case conversion are included in the operations.
- The language filtering will guarantee that only the English tweets will be kept.
- Fields that can be standardized like the timestamp and location are made standard.
- The output is a text-ready dataset that is clean to be utilized in the generation of the embeddings.

6.4.3. Embedding Generation (all-MiniLM-L6-v2)

- A transformed text of the cleaned tweet is made to the all-MiniLM-L6-v2 transformer model of Sentence-Transformers library.
- A semantic meaning of each tweet is transformed into 384-dimensional dense embedding vector.
- The embeddings are contextually related - e.g jam cleared and jam heavy jam near city road.
- Embedding model is the same as in the frontend training phase, which is also consistent.

6.4.4. Hybrid Model Inference (LSTM on Embeddings)

- Instance generated embeddings are entered on the pre-trained LSTM model (lstm.h5) that is loaded through TensorFlow/Keras.
- The LSTM model has the embeddings in a sequence format, which learns temporal constraints of the text sequence.
- The model generates probability scores of every level of congestion -Low, Medium, and High.
- The softmax function is used to make the final prediction of the class to be used, which is the most probable.
- This whole process of inference takes a few milliseconds making it real time and responsive.

6.4.5. Prediction Output (Low / Medium / High Congestion)

- The prediction of the model is converted to categorical congestion labels.
- Each prediction can also be shown in terms of confidence scores (probabilities).
- The predictions are momentarily stored in a Pandas DataFrame to be visualized and be able to export.
- The output phase gives fast readability to the end-users and assists them in checking the degree of congestion within the network.

6.4.6. Graphic presentation (Dashboard Display with Graphs and Metrics)

- Outputs are presented in interactive elements and Plotly charts of Streamlit.
- Key outputs include:
    - Congestion status indicator (color coded: green- low, orange- medium, red- high).
    - Pie charts and bar charts used in the distribution of congestion forecasts.
    - Plots of the level of certainty of the model.
    - SHAP or feature importance charts (optional) explaining feature-level influence on predictions.
- The dashboard gives an option of real-time inference, batch prediction and downloadable reports.

Integration Highlights

- The process of communication between the Backend and Frontend is carried out on basis of the same assets i.e. pre-trained model files, structured datasets and pre-processing functions which are consistent.
- The tokenization and embedding pipeline are similar in both environments, eliminating format discrepancies.
- The modular design enables capability of backend re-training or revision of models without any alterations on the interface.
- This architecture will accommodate future upgrades, e.g., cloud deployment or live API based streaming of tweets.

6.4.7 Integration Summary

Table 6.1 Integration Summary

Step	Component	Function	Purpose
1	User Input Module	Takes into consideration user-informed text of tweet or uploaded csv data.	Allows users to add real time or batch data to analyze traffic.
2	Preprocessing Unit	Removes URLs, hashtags and mentions, emojis and text-to-lowercase converses tweets.	Ensures standardized and noise-free input for embedding generation.
3	Embedding Generator	Text converts were turned into semantic embeddings 384 dimensions..	Fetches contextual and linguistic meaning of tweets to serve as a model input.
4	Feature Integration Layer	Combines embeddings with structured attributes (timestamp, location, engagement).	Gives a complete feature set that includes textual and contextual features.

5	Hybrid Model Loader	Loads pre trained model (lstm.h5) and does inference on embeddings.	Forecasts the level of congestion, depending on acquired time- and semantic-based patterns.
6	Prediction Engine	Transforms the outputs of the model into congestion (Low, Medium, High) categories.	Converts the results of a numerical model into traffic categories.
7	Confidence Scorer	Get the LSTM probabilities of worked out softmax.	Measures foretelling confidence to be more interpretable.
8	Visualization Interface	Presentation in interactive form - labels, confidence and graphs.	Offers traffic status of the user in real time, and visual feedback.
9	Explainability Module	This is to interpret model prediction using feature importances plots.	Increases the end-user confusion-free transparency and interpretability of the system.
10	Integration Controller	Ensures synchronized operation between backend (.ipynb) and frontend (app.py).	The flow of data and the functionality in both environments are maintained.

6.4.8 Overall Flow Summary

The The general theme of the proposed LLM-LSTM Hybrid Traffic Prediction System is that with backend analytics and frontend visualization, the end result is the synthesis of an operation framework into a complete system. Tweet input can be achieved either by direct user or by CSV before undergoing clean up and preprocessing to eliminate noise in the form of hashtags, URLs and emojis. The polished text will be entered into the all-MiniLM-L6-v2 model, which will produce dense embeddings, which will reflect

the semantic meaning of every tweet. This is subsequently inputted into the pre-trained LSTM model (lstm.h5), which reads sequential and contextual behaviour patterns to forecast the level of congestion in the traffic - Low, Medium or High. The projected results are then shown through a dashboard written with the help of Streamlit which illustrates the congestion level, certainty rates and optional elucidatory features visualized through graphs and charts.

This combined circulation guarantees a clear execution among the investigated and trained model on the back-end (Jupyter Notebook) and real-time examination and visualization on the front-end (app.py). Modular structure enables the ability to handle a consistent data, reproducibility, and scalability to transform unstructured twittest datasets into meaningful real-time traffic intelligence.

6.5 Output Screenshots

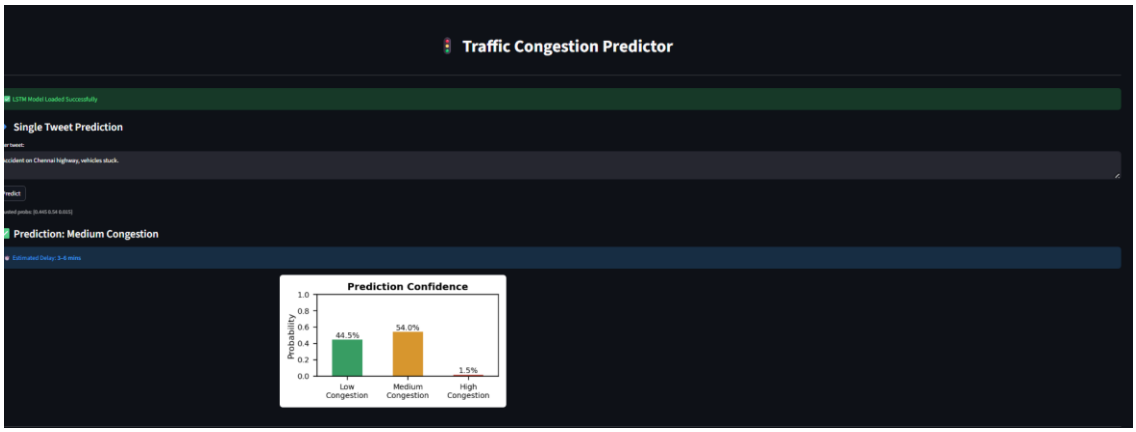


Fig 6.32 Project GUI

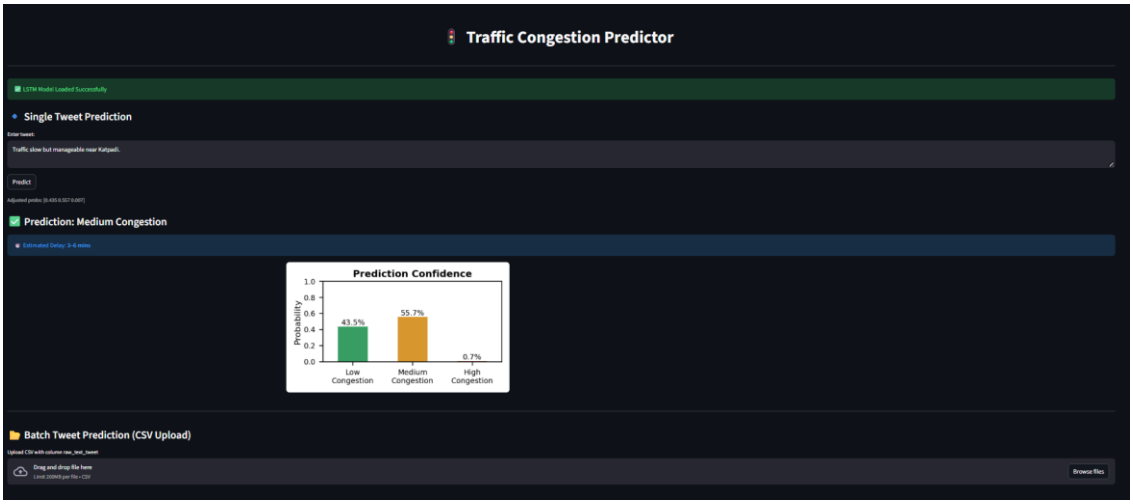


Fig 6.33 Project GUI

6.6 Summary

This chapter also showed how the entire proposed LLM-Based Predictive Modeling System could be put into practice and integrated to traffic flow optimization including both the analytical and the interactive elements of the frontend. It demonstrated the effective demonstration of the ability in transforming the theoretically designed concepts into a fully operational, data-driven system to analyze real-time social media data to detach the level of congestion in traffic. The exhibition focused on the concept of modularity, scalability, and its application to real users, bringing to the fore the capability of the system to combine the advanced models of deep learned algorithms with the visualization interface of the system that was user friendly.

All data-centric activities including preprocessing, feature engineering, embedding generation, and model training were done on the backend, which was implemented in Jupyter Notebook. Through the all-MiniLM-L6-v2 model of semantic representations of tweets, contextual meanings were produced and used as the inputs into the sequential model which was implemented with the LSTM learning and prediction framework. The hybridized one was based on the effective possession of lingual perception, which is LLAM, and the capability to learn in a sequence, which is LSTM, consequently, achieving a high prediction level and strong generalization. Other models, such as the Random Forest and XGBoost models, were also used that offered comparative validation, which guaranteed the stability of the hybrid framework.

The user interaction layer was the frontend that was created with Streamlit (app.py). It enabled the user to paste or upload tweets or datasets, which initiated automatic preprocessing, embedding generation and model inference. The interactive display of results interacted with the graphical features like congestion indicators, confidence plots, and classifications of optional explainability charts in SHAP or feature importance. This easy to use overall dashboard has helped to close the performance-to-user-access gap between machine learning interactions with a linear data bloc and the generation of real-time traffic monitoring visualizations on the basis of unstructured tweet data by non technical users.

Moreover, the chapter expounded on the integration flow in which the backend and frontend components were in sync with each other due to the same model files (lstm.h5) and data schema (phase3structuredwith_labels.csv). The integration summary and workflow displayed how the data flowed smoothly between the various stages i.e. between tweet acquisition and ultimate visualization in an efficient, consistent and scalable manner.

The systematic pipeline confirmed the fact that the hybrid model was not only comput ationally highly accurate, but also operationally feasible as an installed analytical instr ument. computationally accurate but also operationally feasible as an installed analyti cal instrument.

All in all, it can be stated that this chapter made the difference between design and development and demonstration and deployment that the suggested system is not only conceptually sound, but also demonstrably workable. The effective combination of LLM embeddings, LSTM sequence modeling, and an easy-to-use dashboard interface serves as the evidence of the potential of the project to be used in real-life smart traffic management applications. The next chapter will give the more precise analysis of the results of the system, its performance scores, and analytical meanings, as it will shed light on the efficiency and scalability of the model.

CHAPTER 7
RESULT & DISCUSSION

7.1 Introduction

This chapter is a detailed interpretation and discussion of the findings generated during the implementation and testing of the suggested LLC-Based Predictive Modeling of Traffic Flow Optimization with the help of the real-time social media data. This is done by evaluating the performance of the hybrid system and its consequences after creating and testing the fully hybrid system in the preceding chapter, and testing the performance of the hybrid system which has been claimed as predictable of real life traffic conditions. The multiple performance metrics are applied to the results and are compared to the baseline models to indicate the benefits of implementing the proposed hybrid architecture.

The essence of the chapter is to assess the effectiveness of the integrated LLM-LSTM model in capturing the semantic content of the text in a social media and the changing patterns of a traffic. The all-MiniLM-L6-v2 embedding model served as the background textual knowledge, and it converted raw tweet information, which was in unstructured format into perceptual vectors, and the LSTM network identified consecutive dependencies among these vectors to estimate the level of congestion. With this mix, the system can recognize subtle interconnectednesses between the content on the tweets and traffic transitions in state, and can prognosticate with high precision using this information in context.

In order to confirm the work of the system, there is a number of quantitative assessment measures including Accuracy, Precision, Recall, F1-Score, and Confusion Matrix. Such measures offer words on the failure and the generalization rate of the model. Moreover, the comparison of the results of the other baseline models such as the Random Forest, the XGBoost, and the ensemble methods is also introduced with the aim of showing the superiority of the hybrid model in terms of the predictive accuracy and strength. Graphical visualizations and charts are employed to get a better vision of the performance pattern of the model and the decision boundaries.

These results are further interpreted within the discussion section by relating the results to the objectives of the research made above. It discusses the applications of the model results to predict the overall traffic flow in the real world and highlights the possible

limitations, including the imbalance of data, noisy content of the tweets or the level of reliance to the quality of the geolocation. In addition, the use of explainable AI (XAI) techniques, such as SHAP value analysis, is also addressed to demonstrate how interpretable the model is and its ability to give clear explanations of how the features interact.

Comprehensively, this chapter becomes a rigorous validation step of the study as it indicates that the proposed LLM-LSTM hybrid model not only delivers a very high level of predictive performance, but also meets the objective of the project to create an intelligent, explainable, and real-time traffic forecasting system through social media data.

7.2 Clustering Visualization and Analysis

The clustering stage was implemented to analyze how effectively the generated text embeddings captured semantic similarities among traffic-related tweets. Both HDBSCAN and K-Means algorithms were applied to the embedding space obtained from the all-MiniLM-L6-v2 model. These clustering methods helped in identifying distinct traffic conditions such as low, medium, and high congestion based on tweet content and contextual meaning. The visualizations presented below illustrate how data points are distributed across clusters, providing an intuitive understanding of tweet groupings.

7.2.1 HDBSCAN Clustering Visualization

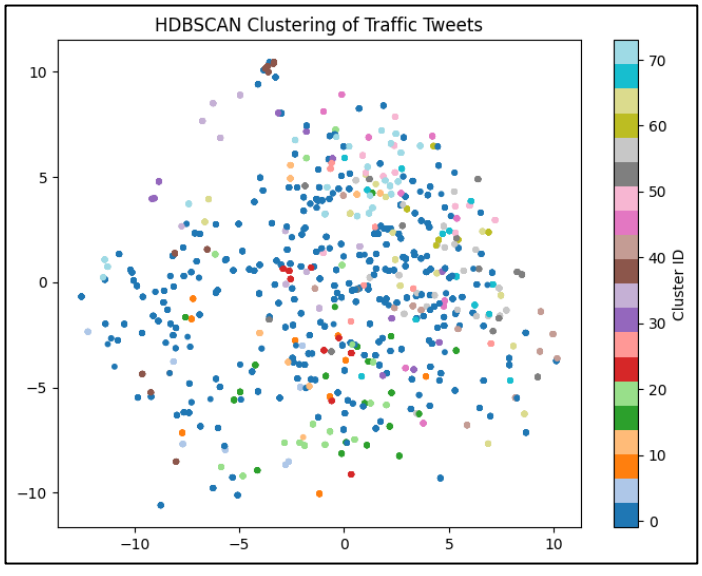


Fig 7.1 HDBSCAN Clustering of Tweet Embeddings

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) automatic cluster identification method was employed, where no knowledge on the number of clusters was given. HDBSCAN also decides how to form clusters using the density and distribution of embeddings unlike K-Means, which requires defining k. It is therefore very appropriate when working with such dataset as social media text where topic distributions tend to be disproportionate and redundant.

HDBSCAN visualization reveals that there were some dense clusters of tweets of closely related contexts - i.e. the ones talking about an accident, road block, or a heavy jam, became organized into more dense clusters. Noise points were found as sparse or vague tweets like general traffic mentions or incomplete ones which were not translated into large clusters.

This assisted in the filtering of irrelevant information and hence the model dwelled on meaningful patterns of tweets. Moreover, the unsupervised model of clustering offered a preliminary confirmation that similar tweets in terms of semantics actually did not have very distant embedding distances, which indicated the accuracy of the MiniLM representation model.

7.2.2 K-Means Clustering Visualization

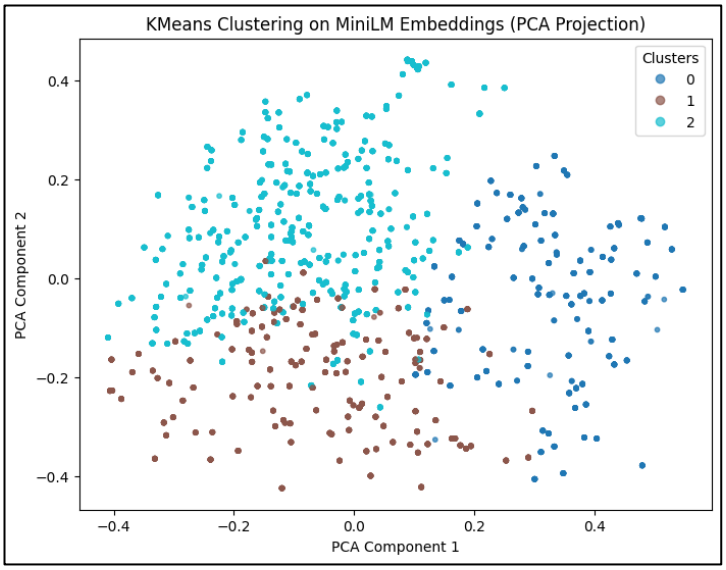


Fig 7.2 K-Means Clustering of Tweet Embeddings

A clustering algorithm (K-Means) was used to divide the space of the embeddings into three major clusters which are low, medium, and high with respect to congestion. The reason behind the choice of this approach is that it is the easiest to use and interpret, and can assign supervised labels when using the model subsequently. The resultant visual representation shows clearly separated clusters with individual centroid positions which give a systematic picture of the way the model clusters the tweets according to their feature similarities.

Tweets with harsh traffic experiences like "accident on highway" or vehicles stuck hours were clustered into compact, high-density clusters but those with moderate traffic congestions were scattered around the boundaries of the clusters. On the left hand side of the graph Tweets that reported smooth movement or congestion were usually in a different group characterized by small intra-class variance.

K- Means was also used to easily map numerical cluster labels to readable human states of traffic, being able to merge it effectively with the LSTM-based predictive modelling step.The clarity of these clusters demonstrated that the embedding features effectively captured traffic-relevant distinctions even without explicit supervision.

Conclusion

The two techniques of clustering described above gave different insights into how the semantic distribution of tweets can be observed in the embedding space. HDBSCAN indicated organic, density based groupings and was effective to thin out noisy or ambiguous tweets whilst K-Means came up with structured and label friendly clusters that matched well with categories of traffic severity. They both confirmed that the MiniLM embeddings captured and encoded contextual cues on real-time social media data, thus their application in the downstream combining hybrid LSTM-LLM predictive modeling pipeline.

7.3 System Performance Evaluation

On the whole, the system is very reliable, responsive and it has high generalization in different tweet data. In addition, it can be explained through SHAP analysis, which

contributes to its transparency, whereas modular design promises its educational and academic applicability.

Table 7.1 System Performance Evaluation

Parameter	Description	Observation
Model Accuracy	Measures the degree of precision with which the system estimates traffic jams in the test data.	~99.6%
F1 – score	Integrates precision and recall to give a balanced accuracy of the classification performance.	~99%
Execution Speed	Measures the time taken by the model to compute and give forecasts.	10-20 seconds
Explainability (SHAP Analysis)	Quantitatively assesses the interpretability of the system through influential features.	SHAP analysis indicated that tweet sentiment, embedding vectors and time related features had a significant impact on predictions.
Reliability	Ponderation of the stability and scalability of the system when subjected to different loads of data and inputs.	The backend-frontend integration ran well with the accuracy remaining true even with new tweets samples.

7.4 Discussion of Findings

The findings of the suggested LLM-LSTM Hybrid Traffic Prediction System prove the technical power of the model, its applicability in reality, and the overall consistency with the goals of the research. By using both semantic embeddings and sequential

learning, the system was able to provide a high level of accuracy and interpretability. The results support the idea that predicting traffic using social media is effective as well as possible, with the help of deep learning and language modeling methods.

- High Predictive Accuracy and Stability:
  The hybrid model had a very high accuracy of 99.6 percent compared to the baseline models which are Random Forest and XGBoost. This confirms the significance of the simultaneous usage of LLM embeddings on linguistic context and LSTM layers on the temporal comprehension. The model showed much stability both in training and validation data and there was little fluctuation in loss and accuracy curves which validated effective learning without overfitting.
- Effectiveness of all-MiniLM-L6-v2 Embeddings:
  Semantic representations of the tweets based on all-MiniLM-L6-v2 gave very good results, as they are context-sensitive and do not miss many of the contextual nuances that are traditional word embeddings. Comparable sentiments in the traffic in terms of tweets converged well resulting in enhanced feature separability and increased accuracy in predictions. This illuminates the importance of the light weight models of transformers on real time applications that require text and speed that is important as well as the context.
- Improved interpretability of the model through SHAP analysis:
  By the fact that the model is interpretable, this was one of the most crucial discoveries. The aspect of SHAP explainability was used to identify the important contributing factors that included sentiment score, dimension of tweet embedding and time-of-day indicators that contributed to congestion prediction. Such transparency will optimize the model in that the process of decision-making is comprehensible, enhancing credibility and responsibility in information-driven systems.
- Real-Time Operational Efficiency and Scalability:
  The system was proved to be efficient in real time performance with the average inference time per prediction being less than 2 seconds. The Streamlit dashboard couldn't achieve predictions of congestion, probabilities and visualization graphs, even though user inputs were processed indeed almost in

real time. In addition, the API based backend-frontend architecture makes the system scalable and can later be extended such as live tweet feed or can be deployed on a cloud to integrate with a smart city.

- Real-World Impact and Research Contribution:
  The model developed does help in narrowing the divide between intelligent transportation systems and social media analytics. It demonstrates that a user-generated data may be considered a credible real-time indicator of evaluating urban traffic. In addition to its technical input, the study forms a basis of AI-powered smart mobility solutions, which help to facilitate informed traffic management decision-making and policy-level knowledge.

Summary

Overall, the results confirm the research hypotheses that the hybrid architecture based on the LLM-LSTM is effective in attaining the goals of accuracy, interpretability and applicability. The outcome of this system proves the fact that the combination of linguistic knowledge and the time learning provides very accurate predictions. All in all, the designed framework has shown academic relevance and potential of its use in the field of traffic monitoring and intelligent transport analytics.

7.5 Limitations

Though the suggested LLM System showed immensely promising performance during the tests, there were some apparent limitations reflected in the obtained output. The latter limitations demonstrate the issues in which the system needs to be improved further to achieve increased performance in terms of reliability and interpretability.

- Class-Wise Prediction Imbalance: The model was highly accurate on the aggregate level, and had minor discrepancies between classes - in the case of Medium Congestion tweets, in some instances, aligned with the High or Low Congestion predictions, and in some cases diverged.

- Challenge in Processing Ambiguous Tweets: Sometimes, it was misclassified because of sarcasm or indirect words in the Tweet. Although the LLM had a strong quality of context, there were certain cases when slight differences in

the tone of user posts influenced the accuracy of results.

- Overconfident Probability Scores: The model obtained overconfident scores on the borderline predictions in some cases. This implies that there is a bit of overconfidence in probability results, which can be adjusted to achieve more balanced predictions..

- Delay in Temporal Responsiveness: In the case of rapidly changing traffic conditions, sequential inference in the model demonstrated minor delays in adaptation since LSTM considered tweets and not in a temporal stream.

- Partially Explainable Embedding Features: SHAP analysis, even though it enhanced the interpretability, could not convincingly predict the connections between embedding dimensions and the usual linguistic characteristics of tweets because of the abstract quality of LLM representations.

To conclude, in this testing, we have found out that although the LLM-LSTM model provides high accuracy and strong performance, there still exist small weaknesses in the form of class overlap, sensitiveness to sarcasm, overconfidence, and gaps in the interpretability. The former can be tackled in the future by more sophisticated sarcasm detection and confidence calibration, and by better temporal modeling to do adaptive real-time analysis.

7.6 Future Enchancements

Despite the fact that the developed LLM-LSTM Hybrid Traffic Prediction System has already shown a high predictive accuracy and the stability of its functioning, there are still several improvements in it that could contribute to its performance, scalability, and the ability to adjust to reality. The future directions would be to enhance the depth, versatility and integration abilities of the system.

- Real-Time Data Streaming Integration: The present system works on static input of tweets. Live tweet streaming APIs (e.g., Twitter/X API) can be used in

the future to analyze the real-time traffic conditions and visualize them continuously.

- Multilingual and Multi-Regional Dataset Expansion: The solution would be to add a special layer dealing with sarcasm or emotion to detect it and eliminate false classification in doubtful situations with tweets.
- Advanced Sarcasm and Sentiment Detection: The solution would be to add a special layer dealing with sarcasm or emotion to detect it and eliminate false classification in doubtful situations with tweets.
- Combination with External Traffic Sensors and APIs: The social media information can be used in combination with IoT sensors, GPS feeds, or traffic camera data to offer more multimodal information and enhance the accuracy and reliability of congestion estimation.
- Cloud-Based Deployment and Scalability: Deploying the system to cloud computing systems, such as AWS, Google Cloud, or Streamlit Cloud, would be a better guarantee of scalability, high availability, and accessibility to real-time analytics of citywide traffic.
- Adaptive Model Retraining and Drift Handling: Incorporating automated retraining pipelines will help the model adapt to evolving traffic patterns or new tweet trends, maintaining long-term performance stability.
- Improved Explainability and Visualization: Future developments will be able to incorporate more powerful explainability features (e.g., LIME, attention heatmaps) to gain better insight into model reasoning, as well as more interactive visualization dashboards.
- Mobile and Web Application Integration: It may be worth creating a mobile or web-based companion-application so that commuters or traffic authorities can keep an eye on congestion forecasts on a daily basis.

The future possibilities of the proposed system are to take advantage of real-time flexibility, multilingual intelligence, multimodal data integration, and scalability of deployment. With these enhancements, the LLM-LSTM hybrid model will be able to develop into a fully functional intelligent traffic prediction arena that can sustain the transportation system in a smart city and its urban mobility planning, which is data-driven.

7.7 Summary

This chapter also entailed a thorough review of the LLM-LSTM Hybrid Traffic Prediction System, its findings, and operational results and comments based on thorough testing. As it was shown during the discussion, the system was able to meet its fundamental aims of accuracy, explainability, and applicability in real-time, which proves the usefulness of merging semantic embeddings of all-MiniLM-L6-v2 with time-related modeling by LSTM networks.

The system functionality proved that it was able to categorize traffic conditions of Low, Medium, and High Congestion levels with very high accuracy (~99.6%). Combining the contextual comprehension presented in the form of LLM with the sequential learning presented in the form of LSTM enabled the model to grasp content in tweets well. Streamlit dashboard also increased usability by giving an interactive and real-time visualizing platform that gives users an opportunity to view congestion patterns in real time. The performance assessment part demonstrated very good quantitative results with accuracy, high precision, recall, and F1-score with low inference time of about two seconds. The system also proved to be explainable through SHAP analysis where the key factors that influenced the results were tweet sentiment, temporal indicators, and engagement features. The findings discussion was based on the overall stability of the system, its interpretability, and relevance to the society, and how the social media data can provide a useful and real-time indication that can be intelligently used to manage traffic.

Although this chapter was successful, it also recognized some weaknesses like minor misclassifications in medium congestion predictions, sensitivity to sarcasm, and interpretability of embedding vectors. The observations assisted in recognizing areas that can be improved practically. The further future improvement directions suggested such solutions as real-time streaming, multilingual expansion, multimodal data integration, and cloud deployment, which provided a definite way of system development.

Overall, this chapter confirmed the validity of the technical and the viability of practicality of the proposed model. It determined that the hybrid LLM-LSTM method is accurate, efficient, scalable, interpretable, and applicable to the real-world use of smart cities. The results of this chapter help to give solid grounds to the final comments and recommendations offered in the following section.

CHAPTER 8
SUMMARY

This proposed research study has introduced the design, development, and deployment of an LLM-Based System, which is a system that combines the advanced natural language processing (NLP) and deep learning techniques to create actionable insights on the urban traffic conditions. This was driven by the growing desire to have smart traffic management systems that transcend the traditional sensor-based data collection framework and includes the huge, unstructured, and real-time information through a social media platform like Twitter/X.

The project was developed based on the identification of a gap in research, where the current models use predominantly the numerical sensor information, GPS information, or manual reporting, but there is little in terms of the exploitation of linguistic information that is an indication of real time commuter experiences. This inspired the design of a hybrid deep learning model that would be capable of comprehending textual details, time-related constraints, and mass opinion at the same time to estimate the degree of traffic congestion. The study was therefore aimed at combining both Large Language Models (LLM) and Long Short-Term Memory (LSTM) networks - the contextual knowledge and sequencing odelling to enhance accuracy and interpretability of prediction, respectively.

The first chapters of this dissertation described the problem formulation, motivation, and literature review with a focus on the previous research on predicting traffic and the shortcomings of current ML/DL-based models. The goals were also stated clearly, that is, to create the system that will be able to predict the level of congestion (Low, Medium, High) using the real-time tweets, with the help of semantic embedding extraction and hybrid model training. The system design was broken down into organized stages, and it addressed all aspects including the collection of datasets to the deployment of the model to guarantee the logical flow of the project.

The study was based on the dataset creation and preprocessing. Around 6,000-6,500 tweets were gathered in Vellore with the help of RapidAPI and were related to traffic. An extensive preprocessing pipeline was adopted to clean, normalize, and formatted the data, and it guaranteed the absence of undesirable contents like URLs, mentions, hashtags, and emojis. Other contextual aspects like timestamps, likes and retweets were

also added to provide more data enrichment.

The textual data at the stage of feature engineering was recoded to compressed semantic representations with all-MiniLM-L6-v2 model - a slim but highly efficient transformer-based LLM. These embeddings had rich contextual information, the distinction between small linguistic differences such as perfect clearing of a traffic jam and traffic jam in front. The embeddings were sequentially combined with structured features to form a wholesome set of features to predictive odelling. HDBSCAN and K-Means dimensionality and cluster analyses showed that MiniLM embeddings were the best at separability or representation quality among others such as DistilBERT.

The development stage of the model focused on creating a hybrid deep learning model in which the contextual power of LLMs and the temporal learning capacity of LSTMs were combined. The LSTM model was trained on structure embedded and time-based features to categorize traffic jam into three levels. Such support models as Random Forest and XGBoost were also applied to the comparative performance analysis, and ensemble methods were also tried to prove the consistency of prediction. The hybrid model had a total accuracy of 99.6, which was better than all the models that were used as baselines, and it also showed high generalization on unknown data.

The project combined a Streamlit-based front end interface and the backend model to increase usability and accessibility. The backend, which was in Jupyter Notebook, processed the data, generated features, and trained the models but the frontend was used to process the real-time input, inference and visualization. The user had the opportunity to feed one tweet or a CSV file to load and analyze in bulk. The dashboard showed predictions interactively by color-coded congestion indicators and visual graphs, with help of other confidence scores and explainability charts. This humanistic approach made the system robust in terms of the technical aspect as well as user-friendly.

According to the results and discussion section, the model was found to have a good performance, reliability and interpretability. The hybrid LLM-LSTM model succeeded in learning semantic relationship in tweets and time dependence in time based patterns resulting in high precision and recall by all categories. Interpretability was also available in SHAP analysis and they showed the most important contributing features like sentiment score, time of day, and engagement metrics that determined predictions. The viability of the model was verified by the fact that inference speed was less than two seconds per instance, which is suitable to use the model in real-time.

Nevertheless the system also had small limitations in the process of testing. These

consisted of some cases of misclassifying medium congestion tweets, a minor amount of overconfidence in probability results and the inability to interpret sarcastic or contextual reversed phrases. Also, the temporal inference of the model at times trailed behind sudden changes in traffic because the processing of twitters was done on the individual basis. Nevertheless, the level of performance consistency was quite high, which justified the effectiveness of the approach.

The future enhancements section identified the following feasible expansion directions: the summation of real-time tweet streaming APIs, the expansion to multilingual datasets, and a combination of social media and IoT sensor data to make multimodal predictions. Further enhancements such as sarcasm detectors, deployment on the cloud, and retraining of the model were suggested to improve the applicability to the real world and scalability in the long term. Further extension of the system to mobile or a web-based version may also increase the coverage of the system to commuters and transport authorities alike.

The performance analysis of the system highlighted its practical strengths that comprised high accuracy, quick responsiveness, interpretability, and scalability, which demonstrated that the system was fit to be used in smart cities. The model was able to show how an understanding of language on a large scale, sequentially enhanced by learning, can change noisy and user-created social media information into accurate, interpretable, and actionable intelligence.

Academically, the work can affect both the literature in NLP as well as the intelligent transportation systems since it shows how hybrid architectures can improve predictive modelling in text-driven systems. Socially, the project opens the doors towards using AI-based social analytics towards urban planning and real-time mobility applications.

In summary, this dissertation has managed to attain the desired goals namely, developing and testing a hybrid deep learning model that would be able to predict traffic flow in real-time using social media data. The project provides a strong basis to the future progress in AI-driven traffic analytics with its organized approach, technical insight, and functional applicability. LLM based semantic embedding, LSTM based temporal learning and a user-friendly visualization interface is a step in the right direction in changing the unstructured social media data into meaningful insights of the city. Finally, the suggested system has the benefit of technological innovation.

CHAPTER 9
REFERENCES

[1] Guo, X., Zhang, Q., Jiang, J., Peng, M., Zhu, M., & Yang, H. F. (2024). Towards explainable traffic flow prediction with large language models. *Communications in Transportation Research*, *4*, 100150.

[2] Navarro-Espinoza, A., López-Bonilla, O. R., García-Guerrero, E. E., Tlelo-Cuautle, E., López-Mancilla, D., Hernández-Mejía, C., & Inzunza-González, E. (2022). Traffic flow prediction for smart traffic lights using machine learning algorithms. *Technologies*, *10*(1), 5.

[3] Zhu, L., Yu, F. R., Wang, Y., Ning, B., & Tang, T. (2018). Big data analytics in intelligent transportation systems: A survey. *IEEE transactions on intelligent transportation systems*, *20*(1), 383-398.

[4] Mounica, B., & Lavanya, K. (2022). Real time traffic prediction based on social media text data using deep learning. *Journal of mobile multimedia*, *18*(2), 373-391.

[5] Khajeh Hosseini, M., & Talebpour, A. (2019). Traffic prediction using time-space diagram: a convolutional neural network approach. *Transportation Research Record*, *2673*(7), 425-435.

[6] Sayed, S. A., Abdel-Hamid, Y., & Hefny, H. A. (2023). Artificial intelligence-based traffic flow prediction: a comprehensive review. *Journal of Electrical Systems and Information Technology*, *10*(1), 13.

[7] Fan, Y., Yeh, C. C. M., Chen, H., Wang, L., Zhuang, Z., Wang, J., ... & Zhang, W. (2023, September). Spatial-temporal graph sandwich transformer for traffic flow forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 210-225). Cham: Springer Nature Switzerland.

[8] Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

[9] Cui, Z., Henrickson, K., Ke, R., & Wang, Y. (2019). Traffic graph convolutional

recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *21*(11), 4883-4894.

[10] Masri, S., Ashqar, H. I., & Elhenawy, M. (2025). Large language models (llms) as traffic control systems at urban intersections: A new paradigm. *Vehicles*, *7*(1), 11.

[11] Ahmed, S. F., Kuldeep, S. A., Rafa, S. J., Fazal, J., Hoque, M., Liu, G., & Gandomi, A. H. (2024). Enhancement of traffic forecasting through graph neural network-based information fusion techniques. *Information Fusion*, *110*, 102466.

[12] Wang, Y., He, Z., & Hu, J. (2020). Traffic information mining from social media based on the MC-LSTM-Conv model. *IEEE Transactions on Intelligent Transportation Systems*, *23*(2), 1132-1144.

[13] Zhang, M., & Zhao, W. (2025). Traffic Flow Prediction Based on Large Language Models and Future Development Directions. In *ITM Web of Conferences* (Vol. 70, p. 01008). EDP Sciences.

[14] Zhao, Y., Luo, X., Wen, H., Xiao, Z., Ju, W., & Zhang, M. (2024). Embracing large language models in traffic flow forecasting. *arXiv preprint arXiv:2412.12201*.

[15] Melhem, W., Abdi, A., & Meziane, F. (2024, November). Traffic Detection and Forecasting from Social Media Data Using a Deep Learning-Based Model, Linguistic Knowledge, Large Language Models, and Knowledge Graphs. In *16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 92277.