**Hybrid Deep Learning for Detecting Novelty Seeking in Travel Reviews: Integrating BERT, CNN, and BiGRU for Improved Accuracy**

**1. INTRODUCTION:**

With the proliferation of powerful sensor embedded smart- phones, crowdsensing has become a leading paradigm which leverages the pervasive smartphone users to collect data efﬁciently. In a typical crowdsensing application, a server posts the required sensing information and recruits a set of smartphone users to collect sensing data. After smartphone users send sensing data to the server, the server aggregates the sensing data to measure phenomena of common interest, i.e., real-time trafﬁc conditions, environmental pollution quality or environmental noise pollution. The accuracy of estimating the common interest depends on the high-quality contributions of highly skilled users. While providing the high-quality contributions, smartphone users consume their energy and the resources of their smart phones such as battery, storage and computing power. In addition, users may expose themselves to potential privacy threats as the sensed data contain time or location tags. Thus, the contributors should be given enough rewards to compensate for their resource consumption or potential privacy leaks. As is known to all, a user wants to maximize her own proﬁt, and may lie or impersonate others to get more payment. Therefore, the design of a secure and truthful incentive mechanism is particularly important. Many incentive mechanisms have been proposed and implemented, such as the reputation systems and monetary approaches. Reputation systems can help identify uncooperative users, but ignore a formal speciﬁcation and analysis of the incentive types and suffer sybil attacks and whitewash attacks. Monetary approaches could be the most promising due to their explicit and ﬂexible incentive methods. Most monetary schemes use pricing strategies to design truthful incentive mechanisms, in which the server and smart phone users cannot increase their utility by cheating or colluding with others. While some other privacy-preserving incentive mechanisms have been proposed for protecting. With the rapid development of information technology, the internet has gradually penetrated many areas of our daily lives. The tourism industry has gradually extended from offline to online. With the emergence of online travel communities, a rapidly increasing number of tourists search the internet for destination introductions and comments about travel experiences from other travelers before making travel decisions. Most online tourism platform reviews reflect what tourists see, feel, and think. Suppose this information is collected and analyzed to visually reveal tourists' praise and criticism attitudes or emotional tendencies about the elements of tourism services. In that case, it will help tourists understand the emotional tendencies of the forerunners towards a certain tourist destination and support tourists in their decision-making. Tour operators can understand tourists' opinions or their attitudes of praise and criticism to maximize their strengths and avoid weaknesses. Reviews also help managers customize products or improve programs and gain a competitive advantage. Personality traits are a group of psychological structures that trigger individual behavior and make individuals respond in the same way to different kinds of stimuli. Traditionally, researchers use self-reporting scales to collect data on personality traits, which requires subjects to self-evaluate their personality traits following the actual situation. Because the personality trait measurement scale mainly relies on the subjects’ subjective feelings and self-statement, most personality trait measurement scales are currently standardized tests. However, individuals responding to surveys are prone to expressing themselves more in line with social values and more conducive to self-representation. In other words, participants can deliberately submit distorted responses, which negatively affects the efficacy of measurement results. Compared with the measurement of personality traits by psychological tests, personality trait recognition based on online behavior data is a method to automatically recognize and judge personality trait types. On the one hand, it overcomes the subjective and static nature of traditional personality trait measurement methods. On the other hand, it also avoids the measurement bias caused by self-reporting and provides new methods and ideas for tourists’ personality traits acquisition. Novelty seeking (NS) is a personality trait, manifested as a general tendency to pursue diversification, curiosity, complexity, and strong feelings and experiences. NS is known to be an important motive for pleasure tourism and is considered an inherent quality. It has been proven to play an inseparable role in the choice of destination and has been one of the greatest impact factors on tourists' perceptions. Previous research has shown that NS affects tourists' return intention, destination loyalty, and satisfaction. NS is a personality trait widely recognized as an influencer of tourism motivation and plays a crucial role in formulating marketing strategies for the tourism industry. Since NS people like to go to remote and unfamiliar places, in the field of personalized recommendation, new tourist destinations can be recommended according to customers' NS tendencies. In addition to developing better recommender systems, organizations can also design more targeted marketing campaigns based on customer needs. It may help improve tourist satisfaction, reduce information duplication and diversify recommendations. From a practical point of view, through NS identification in online travel reviews, user groups with NS characteristics can be accurately identified and located. However, due to the increasing amount of information in tourism online reviews, identifying tourists' NS characteristics is a difficult task. Manually analyzing a large number of online reviews is a time-consuming and costly method for identifying NS. To address this issue, this study attempts to introduce deep learning methods. After developing a multidimensional NS scale, we apply deep learning named BERT-BiGRU model to detect and classify NS in online travel reviews. In the next section, this study reviews the relevant literature. Then, a deep learning-based NS recognition model was developed and tested. Finally, the application of the model is discussed. This study also discusses implications for future research and practice.

**1.1 Objective of the project:**

Understanding the inherent personality trait of novelty seeking (NS) plays a crucial role in tourism motivation and destination selection. However, the manual classification of travel reviews, such as those found on TripAdvisor, proves challenging due to the unstructured nature and high volume of data. This paper proposes a deep learning-based framework for automatically classifying reviews based on the NS personality trait, which is divided into four dimensions: relaxation seeking, experience seeking, arousal seeking, and boredom alleviation. Using a dataset of 30,000 reviews, a model combining Bidirectional Encoder Representations from Transformers (BERT) and Bidirectional Gated Recurrent Unit (BiGRU) was developed. The classifier achieved high precision (93.4%) and F1 scores (93.3%), demonstrating the effectiveness of this model in recognizing NS from travel reviews. The results suggest that BERT-BiGRU can accurately capture and classify NS personality traits, offering practical applications in tourism marketing and recommendation systems.

**2. LITERATURE SURVEY:**

# IoT Security and Privacy Using Deep Learning Model: A Review

Today, the worldwide Internet of Things is one of the emerging technologies and very useful for all smart handling things like (home, smart meter, parking, infrastructure surveillance, healthcare, and government, maritime, banking, smart communication, smart vehicles) etc This makes our life easier every day. IoT is a centralized device that anyone can access, but it is still vulnerable to security attacks such as botnets, DDoS/DoS, malicious attacks, and Shinhole attacks, which IoT must address. These issues include the use of a large number of heterogeneous devices, which compromises security and scalability, the need for more energy efficient devices, and the centralized system that anyone can access, making it easy for attackers. The aim of this paper is to provide a comprehensive discussion of deep learning algorithms and standard dataset in IoT for security and privacy concerns. Our primary focus is on enhancing IoT security through deep learning. First, we look at deep learning algorithms and classification in IoT security from the standpoint of device design and methodologies. Second, we examine the suitability of IoT systems in terms of security.

# Age and gender differences in online travel reviews and user-generated-content (UGC) adoption: extending the technology acceptance model (TAM) with credibility theory

This study examines the effects of trustworthiness, expertise, perceived usefulness (PU), and perceived ease of use (PEOU) on usage intention toward user-generated content (UGC) and online reviews among female and male younger and older travelers using SEM. The study tested the model with a sample of 200 UK residents who had taken at least one leisure trip in the preceding 12 months and searched for travel information in advance on travel-review websites. PU was the strongest determinant of UGC usage for males, but it failed to be significant among females. PEOU was the strongest determinant of females’ and older travelers’ usage, but it was non-significant for males and younger travelers. expertise had a significant influence on younger travelers, but not older ones. The results provide additional contributions on the effects of gender and age in online travel reviews to help advance both theory and practice.

# Sport Tourism: A Critical Analysis of Research

The concept of sport related tourism has become more prominent in the last few years both as an academic field of study and an increasingly popular tourism product. The purpose of this paper is to review and critique the sport tourism literature as it stands in 1998, and to suggest a future research agenda. Disparities in the definition of sport tourism are addressed and some of the difficulties which scholars have faced in establishing a standardized definition are outlined. In answering the question why has sport tourism suddenly become so prominent, a look back at history shows that people have engaged in sport related travel for centuries. However, in the past ten years, the popularity of this form of travel has increased. Various explanations, such as the increased emphasis on health and fitness and increased use of sports events by cities to attract tourists, are examined. The question of what is known about sport tourism includes a review and critique of the literature in the three domains of sport tourism: active sport tourism, which refers to people who travel to take part in sport; event sport tourism, which refers to travel to watch a sports event; and nostalgia sport tourism, which includes visits to sports museums, famous sports venues, and sports themed cruises. The overarching conclusion from this review is that the field suffers from a lack of integration in the realms of policy, research, and education. At a policy level, there needs to be better coordination among agencies responsible for sport and those responsible for tourism. At a research level, more multi-disciplinary research is needed, particularly research which builds upon existing knowledge bases in both sport and tourism. In the realm of education, territorial contests between departments claiming tourism expertise and those claiming sport expertise need to be overcome.

# Tourist behavior and personality

Stanley Plog's concept of psychocentric -allocentric in travel preferences was operationally measured by two scales: one of preferred destinations and another of preferred activities while on vacation. It was found, for females, that psychocentric (who are non-adventurous) were less neurotic but more extraverted than allocentric.

# Rude customers and service performance: roles of motivation and personality

On the basis of the self-determination theory, we develop and test an integrative framework that explains when and why customer incivility impairs employee service performance. Using multisource data collected through two waves in a shopping mall, we found that the strength of the mediated relationship between customer incivility and employee service performance (via employee intrinsic motivation) varied based on employee core-self evaluations; the negative indirect effect of customer incivility via intrinsic motivation on service performance was weaker for employees with high levels of core-self evaluations than for employees with low levels of core-self evaluations.

# A Systematic Review and Meta-Analysis of Psychological Research on Conspiracy Beliefs: Field Characteristics, Measurement Instruments, and Associations With Personality Traits

In the last decade, the number of investigations of the beliefs in conspiracy theories has begun to increase in the fields of social, differential, and experimental psychology. A considerable number of variables have been suggested as predictors of conspiracy beliefs, amongst them personality factors such as low agreeableness (as disagreeableness is associated with suspicion and antagonism) and high openness to experience (due to its positive association to seek out unusual and novel ideas). The association between agreeableness, openness to experience and conspiracy beliefs remains unclear in the literature. The present study reviews the literature of psychological studies investigating conspiracy beliefs. Additionally, the association between Big Five personality factors and conspiracy beliefs is analyzed meta-analytically using random-effects models. Ninety-six studies were identified for the systematic review. A comprehensive account of predictors, consequences, operationalization, questionnaires, and most prominent conspiracy theories is presented. For meta-analysis, 74 effect sizes from 13 studies were extracted. The psychological literature on predictors of conspiracy beliefs can be divided in approaches either with a pathological (e.g., paranoia) or socio-political focus (e.g., perceived powerlessness). Generally, there is a lack of theoretical frameworks in this young area of research. Meta-analysis revealed that agreeableness, openness to experience, and the remaining Big Five personality factors were not significantly associated with conspiracy beliefs if effect sizes are aggregated. Considerable heterogeneity in designs and operationalization characterizes the field. This article provides an overview of instrumentation, study designs, and current state of knowledge in an effort toward advancement and consensus in the study of conspiracy beliefs.

# Measurement and research using the Big Five, HEXACO, and Narrow Traits: a primer for researchers and practitioners

Objective Personality traits influence human behaviour across a broad range of situations and are consequently relevant to many theoretical and applied disciplines. In this perspective piece, we provide an overview of the logic underpinning personality measurement and review major personality taxonomies. We provide an extensive set of recommendations for researchers and practitioners on when and how to use measures of personality traits. Method We overview a range of taxonomic representations of personality structure focusing particularly on hierarchical representations and five and six factor models such as the Big Five and HEXACO models. We review the various strengths and weaknesses of each approach. Results The review outlines the major reasons for the dominance of the Big Five model, and suggests it is a good descriptive framework for studying personality in general. However we suggest that researchers and practitioners also consider alternative taxonomic personality representations such as the HEXACO. We provide a range of scenarios whereby alternative frameworks will be more appropriate than the Big Five and offer recommendations both for choosing measures in general and for implementing studies examining personality facets. Conclusion Whilst the Big Five represents an excellent general personality framework that is appropriate across multiple situations, researchers and practitioners should be aware of alternative measures and utilise them where appropriate

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM:**

The existing system uses a deep learning model based on Bidirectional Encoder Representations from Transformers (BERT) combined with Bidirectional Gated Recurrent Unit (BiGRU) to classify travel reviews and automatically recognize the NS personality trait. The classification framework is based on four NS dimensions: relaxation seeking, experience seeking, arousal seeking, and boredom alleviation. This model was tested on a dataset of 30,000 TripAdvisor reviews, achieving a precision score of 93.4% and an F1 score of 93.3%. The approach demonstrates strong performance in recognizing personality traits from travel reviews and provides a foundation for applications in tourism marketing and recommendation systems.

**Disadvantages:**

1. The existing system uses only a combination of BERT and BiGRU, which may not fully capture all relevant features in the dataset.
2. The performance could be enhanced by integrating additional algorithms to further optimize feature extraction and improve classification accuracy.
3. The model may be limited in addressing complex interactions within the data that could improve the precision of the predictions.
4. The system relies on a fixed combination of two algorithms, which may not adapt as well to new or evolving datasets.

**3.2 PROPOSED SYSTEM:**

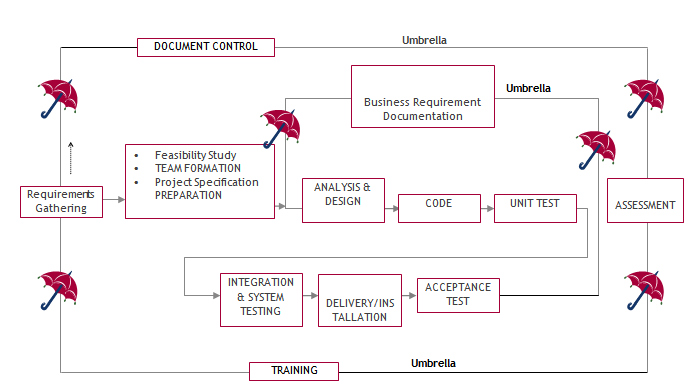
In the proposed system, a hybrid model combining three different deep learning algorithms — BERT, Convolutional Neural Networks (CNN), and BiGRU — is introduced. This model is designed to extract more relevant features from the travel review dataset, improving overall prediction accuracy. By combining BERT's ability to handle contextual language understanding, CNN's capacity for feature extraction, and BiGRU's sequential processing strengths, the hybrid model enhances the classifier's performance, achieving an accuracy of 95% or higher. The integration of these three algorithms enables the model to capture a broader range of features, providing more accurate insights into the NS personality trait.

**Advantages:**

1. Increased accuracy: The hybrid model of BERT, CNN, and BiGRU improves accuracy, achieving 95% or more in classifying personality traits from travel reviews.
2. Better feature extraction: The combination of CNN and BiGRU with BERT allows the model to capture more relevant and complex features from the data, enhancing performance.
3. Improved handling of diverse data: The hybrid approach provides better adaptability and robustness to variations in the data, making the model more effective across different review styles and contents.
4. Enhanced prediction reliability: By using multiple algorithms in tandem, the model is better equipped to handle the intricate nuances of the dataset, providing more reliable predictions for tourism marketing and recommendation systems.
5. Comprehensive feature understanding: The hybrid model captures both contextual understanding and complex sequential patterns in the data, resulting in a more holistic recognition of NS traits.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**



SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_(business)) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user-friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB (min)
* Hard Disk - 500 GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python (3.7.0)

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**SEQUENCE DIAGRAM:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.



**COMPONENT DIAGRAM:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**ACTIVITY DIAGRAM:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

Upload Database

Preprocess & Split Dataset

Train BERT-LSTM algorithm

Predict

Train BERT-Bi-GRU algorithm

Train BERT-CNN-Bi-GRU algorithm

Comparision Graph

User

System

**Data flow:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. Upload Database 2. Successfully Upload Database

3. Preprocess & Split Dataset 4. Successfully Preprocess & Split Dataset

5. train BERT-LSTM algorithm 6. Successfully train BERT-LSTM algorithm

7. train BERT-Bi-GRU algorithm 8. Successfully train BERT-Bi-GRU algorithm

9. train BERT-CNN-Bi-GRU 10. Successfully train BERT-CNN-Bi-GRU

algorithm algorithm

11. Performance graph 12. Successfully performance graph

13. Predict 14. Successfully Predicted

**5. IMPLEMETATION**

**5.1 PYTHON**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**5.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**5.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**5.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI applicatioMakeske the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**5.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**5.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**Algorithms Used:**

**BERT,**

**It is an acronym** **for Bidirectional Encoder Representations from Transformers**, stands as an open-source **machine learning framework**designed for the realm of **natural language processing (NLP)**. Originating in 2018, this framework was crafted by researchers from Google AI Language. The article aims to explore the **architecture, working and applications of BERT**.

## What is BERT?

BERT (Bidirectional Encoder Representations from Transformers) leverages a transformer-based neural network to understand and generate human-like language. BERT employs an encoder-only architecture. In the original Transformer architecture, there are both encoder and decoder modules. The decision to use an encoder-only architecture in BERT suggests a primary emphasis on understanding input sequences rather than generating output sequences.

### Bidirectional Approach of BERT

Traditional language models process text sequentially, either from left to right or right to left. This method limits the model’s awareness to the immediate context preceding the target word. BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

### Pre-training and Fine-tuning

The BERT model undergoes a two-step process:

1. Pre-training on Large amounts of unlabeled text to learn contextual embeddings.
2. Fine-tuning on labeled data for specific [NLP](https://www.geeksforgeeks.org/natural-language-processing-overview/) tasks.

#### Pre-Training on Large Data

* BERT is pre-trained on large amount of unlabeled text data. The model learns contextual embeddings, which are the representations of words that take into account their surrounding context in a sentence.
* BERT engages in various unsupervised pre-training tasks. For instance, it might learn to predict missing words in a sentence (Masked Language Model or MLM task), understand the relationship between two sentences, or predict the next sentence in a pair.

#### Fine-Tuning on Labeled Data

* After the pre-training phase, the BERT model, armed with its contextual embeddings, is then fine-tuned for specific natural language processing (NLP) tasks. This step tailors the model to more targeted applications by adapting its general language understanding to the nuances of the particular task.
* BERT is fine-tuned using labeled data specific to the downstream tasks of interest. These tasks could include sentiment analysis, question-answering, [named entity recognition](https://www.geeksforgeeks.org/named-entity-recognition/), or any other NLP application. The model’s parameters are adjusted to optimize its performance for the particular requirements of the task at hand.

BERT’s unified architecture allows it to adapt to various downstream tasks with minimal modifications, making it a versatile and highly effective tool in [natural language understanding](https://www.geeksforgeeks.org/nlp-vs-nlu-vs-nlg/) and processing.

**5.2 Sample Code:**

**Main.py**

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from string import punctuation

import numpy as np

import pandas as pd

import pickle

#importing NLP packages for text review processing

from nltk.corpus import stopwords

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import accuracy\_score

import os

from sentence\_transformers import SentenceTransformer #loading bert sentence model

import nltk

from nltk.stem import WordNetLemmatizer

from nltk.stem import PorterStemmer

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential, Model, load\_model

from keras.models import model\_from\_json

from keras.utils.np\_utils import to\_categorical

from keras.callbacks import ModelCheckpoint

from keras.layers import Bidirectional, GRU, LSTM, Conv1D, MaxPooling1D, RepeatVector#loading GRU, bidriectional, lstm and CNN

from sklearn.model\_selection import train\_test\_split

import seaborn as sns

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

main = tkinter.Tk()

main.title("Hybrid Deep Learning for Detecting Novelty Seeking in Travel Reviews: Integrating BERT, CNN, and BiGRU for Improved Accuracy") #designing main screen

main.geometry("1300x1200")

global filename, dataset, X\_train, X\_test, y\_train, y\_test, X, Y, scaler, pca,bert,index

global accuracy, precision, recall, fscore, values,cnn\_model,extension\_model,predict,values

precision = []

recall = []

fscore = []

accuracy = []

bert = SentenceTransformer('nli-distilroberta-base-v2')

print("Bert model initialized")

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

ps = PorterStemmer()

def cleanText(doc):

tokens = doc.split()

table = str.maketrans('', '', punctuation)

tokens = [w.translate(table) for w in tokens]

tokens = [word for word in tokens if word.isalpha()]

tokens = [w for w in tokens if not w in stop\_words]

tokens = [word for word in tokens if len(word) > 1]

tokens = [ps.stem(token) for token in tokens]

tokens = [lemmatizer.lemmatize(token) for token in tokens]

tokens = ' '.join(tokens)

return tokens

def uploadDataset():

global filename, dataset, labels, values

filename = filedialog.askopenfilename(initialdir = "Dataset")

text.delete('1.0', END)

text.insert(END,'Dataset loaded\n\n')

dataset = pd.read\_csv(filename)

labels, count = np.unique(dataset['label'], return\_counts = True)

labels = ['Novelty Seeking', 'Non-Novelty Seeking']

height = count

bars = labels

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xlabel("Dataset Class Label Graph")

plt.ylabel("Count")

plt.show()

def processDataset():

global dataset, X, Y

global X\_train, X\_test, y\_train, y\_test, pca, scaler

text.delete('1.0', END)

if os.path.exists("model/bert.npy"):

X = np.load("model/bert.npy")

Y = np.load("model/label.npy")

else:

textdata = []

labels = []

for i in range(len(dataset)):#loop all reviews from dataset

msg = dataset.get\_value(i, 'content')#read review content

label = dataset.get\_value(i, 'label')#read label

msg = msg.strip().lower() #convert text to lower case

msg = cleanPost(msg)#clean the review message

textdata.append(msg)#add message to textdata array

labels.append(label)#adding label to array

embeddings = bert.encode(textdata, convert\_to\_tensor=True)#convert all text data into BERT vector

X = embeddings.numpy()#convert bert vector into numpy for training

np.save("model/bert", X)#save bert data and labels to model folder

Y = np.asarray(labels)

np.save("model/label", Y)

text.insert(END,"Bert Converted Embedding vector from dataset reviews"+"\n")

text.insert(END,X)

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

X = np.reshape(X, (X.shape[0], 32, 24))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

print()

text.insert(END,"\n\nDataset train & test split as 80% dataset for training and 20% for testing"+"\n")

text.insert(END,"Training Size (80%): "+str(X\_train.shape[0])+"\n") #print training and test size

text.insert(END,"Testing Size (20%): "+str(X\_test.shape[0])+"\n")

print()

def calculateMetrics(algorithm, predict, testY):

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

print()

text.insert(END,algorithm+' Accuracy : '+str(a)+"\n")

text.insert(END,algorithm+' Precision : '+str(p)+"\n")

text.insert(END,algorithm+' Recall : '+str(r)+"\n")

text.insert(END,algorithm+' FMeasure : '+str(f)+"\n")

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

def trainLSTM():

global X\_train, y\_train, X\_test, y\_test

global accuracy, precision, recall, fscore,predict,index,values

text.delete('1.0', END)

lstm = Sequential() #defining deep learning sequential object

#adding GRU layer with 32 filters to filter given input X train data to select relevant features

lstm.add(LSTM(32, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True))

#adding dropout layer to remove irrelevant features

lstm.add(Dropout(0.3))

#adding another layer

lstm.add(LSTM(32))

lstm.add(Dropout(0.3))

#defining output layer for prediction

lstm.add(Dense(y\_train.shape[1], activation='softmax'))

#compile GRU model

lstm.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

#start training model on train data and perform validation on test data

if os.path.exists("model/lstm\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/lstm\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = lstm.fit(X\_train, y\_train, batch\_size = 16, epochs = 35, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/lstm\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

lstm = load\_model("model/lstm\_weights.hdf5")

predict = lstm.predict(X\_test)#perfrom prediction on test data

predict = np.argmax(predict, axis=1)

target = np.argmax(y\_test, axis=1)

#calculate accuracy and other metrics

calculateMetrics("BERT-LSTM Model", predict, target)

def trainBILSTM():

global X\_train, y\_train, X\_test, y\_test

global accuracy, precision, recall, fscore,predict,values

text.delete('1.0', END)

gru\_bilstm = Sequential() #defining deep learning sequential object

#adding GRU layer with 32 filters to filter given input X train data to select relevant features

gru\_bilstm.add(Bidirectional(GRU(32, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True)))

#adding dropout layer to remove irrelevant features

gru\_bilstm.add(Dropout(0.3))

#adding another layer

gru\_bilstm.add(Bidirectional(GRU(32)))#adding bidirectional-GRU layer and peform training on X\_train Bert data

gru\_bilstm.add(Dropout(0.3))

#defining output layer for prediction

gru\_bilstm.add(Dense(y\_train.shape[1], activation='softmax'))

#compile GRU model

gru\_bilstm.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

#start training model on train data and perform validation on test data

if os.path.exists("model/bigru\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/bigru\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = gru\_bilstm.fit(X\_train, y\_train, batch\_size = 16, epochs = 35, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/bigru\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

gru\_bilstm = load\_model("model/bigru\_weights.hdf5")

predict = gru\_bilstm.predict(X\_test)#perfrom prediction on test data

predict = np.argmax(predict, axis=1)

target = np.argmax(y\_test, axis=1)

#calculate accuracy and other metrics

calculateMetrics("BERT-Bi-GRU Model", predict, target)

def runCNN():

global X\_train, y\_train, X\_test, y\_test,extension\_model

global accuracy, precision, recall, fscore,predict,index,values

text.delete('1.0', END)

extension\_model = Sequential()

#defining CNN layer

extension\_model.add(Conv1D(filters=32, kernel\_size = 15, activation = 'relu', input\_shape = (X\_train.shape[1], X\_train.shape[2])))

extension\_model.add(Conv1D(filters=16, kernel\_size = 12, activation = 'relu'))

#adding maxpool layer

extension\_model.add(MaxPooling1D(pool\_size = 2))

extension\_model.add(Dropout(0.3))

extension\_model.add(Flatten())

extension\_model.add(RepeatVector(2))

#adding bidirectional + GRU to CNN layer

extension\_model.add(Bidirectional(GRU(24, activation = 'relu')))

extension\_model.add(Dropout(0.3))

#defining output layer

extension\_model.add(Dense(units = 33, activation = 'softmax'))

extension\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))

#compile and train the model

extension\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

if os.path.exists("model/extension\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/extension\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = extension\_model.fit(X\_train, y\_train, batch\_size = 16, epochs = 35, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/extension\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

extension\_model = load\_model("model/extension\_weights.hdf5")

#perform prediction on test data using bidirectional LSTM on test data

predict = extension\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

target = np.argmax(y\_test, axis=1)

#calculate accuracy and other metrics

calculateMetrics("Propose BERT-CNN-Bi-GRU Model", predict, target)

def graph():

global accuracy, precision, recall, fscore

text.delete('1.0', END)

df = pd.DataFrame([['BERT-LSTM','Precision',precision[0]],['BERT-LSTM','Recall',recall[0]],['BERT-LSTM','F1 Score',fscore[0]],['BERT-LSTM','Accuracy',accuracy[0]],

['BERT-Bi-GRU','Precision',precision[1]],['BERT-Bi-GRU','Recall',recall[1]],['BERT-Bi-GRU','F1 Score',fscore[1]],['BERT-Bi-GRU','Accuracy',accuracy[1]],

['Propose BERT-CNN-Bi-GRU','Precision',precision[2]],['Propose BERT-CNN-Bi-GRU','Recall',recall[2]],['Propose BERT-CNN-Bi-GRU','F1 Score',fscore[2]],['Propose BERT-CNN-Bi-GRU','Accuracy',accuracy[2]],

],columns=['Parameters','Algorithms','Value'])

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')

plt.title("All Algorithms Performance Graph")

plt.show()

def predict():

global X\_train, y\_train, X\_test, y\_test,predict,extension\_model

global accuracy, precision, recall, fscore

text.delete('1.0', END)

testData = pd.read\_csv("Dataset/testData.csv")#reading test data

for i in range(len(testData)):

msg = dataset.get\_value(i, 'content')#read test review as message

msgs = msg.strip().lower() #convert to lower case

msgs = cleanText(msgs)#clean messages

data = []

data.append(msgs)#add message to array

embeddings = bert.encode(data, convert\_to\_tensor=True)#convert message review to bert vector

X = embeddings.numpy()#convert vector to numpy

X = np.reshape(X, (X.shape[0], 32, 24))#reshape vector

predict = extension\_model.predict(X)#using extension model predict weather test message is Novelty or not

predict = np.argmax(predict)

text.insert(END,"Text Review : "+msg+" ===> Predicted As "+labels[predict]+"\n")

font = ('times', 16, 'bold')

title = Label(main, text='Hybrid Deep Learning for Detecting Novelty Seeking in Travel Reviews: Integrating BERT, CNN, and BiGRU for Improved Accuracy')

title.config(bg='gray24', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=27,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=200)

text.config(font=font1)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Attack Database", command=uploadDataset)

uploadButton.place(x=10,y=100)

uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess & Split Dataset", command=processDataset)

processButton.place(x=220,y=100)

processButton.config(font=font1)

svmButton = Button(main, text="Run BERT-LSTM", command=trainLSTM)

svmButton.place(x=450,y=100)

svmButton.config(font=font1)

knnButton = Button(main, text="Run BERT-Bi-GRU", command=trainBILSTM)

knnButton.place(x=600,y=100)

knnButton.config(font=font1)

dtButton = Button(main, text="Run Propose BERT-CNN-Bi-GRU", command=runCNN)

dtButton.place(x=770,y=100)

dtButton.config(font=font1)

xgButton = Button(main, text="Comparision Graph", command=graph)

xgButton.place(x=10,y=150)

xgButton.config(font=font1)

dnnButton = Button(main, text="Predict", command=predict)

dnnButton.place(x=200,y=150)

dnnButton.config(font=font1)

main.config(bg='gold')

main.mainloop()

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifying as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property function as a unit. The test data should be chosen such that it passed through all possible condition. Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

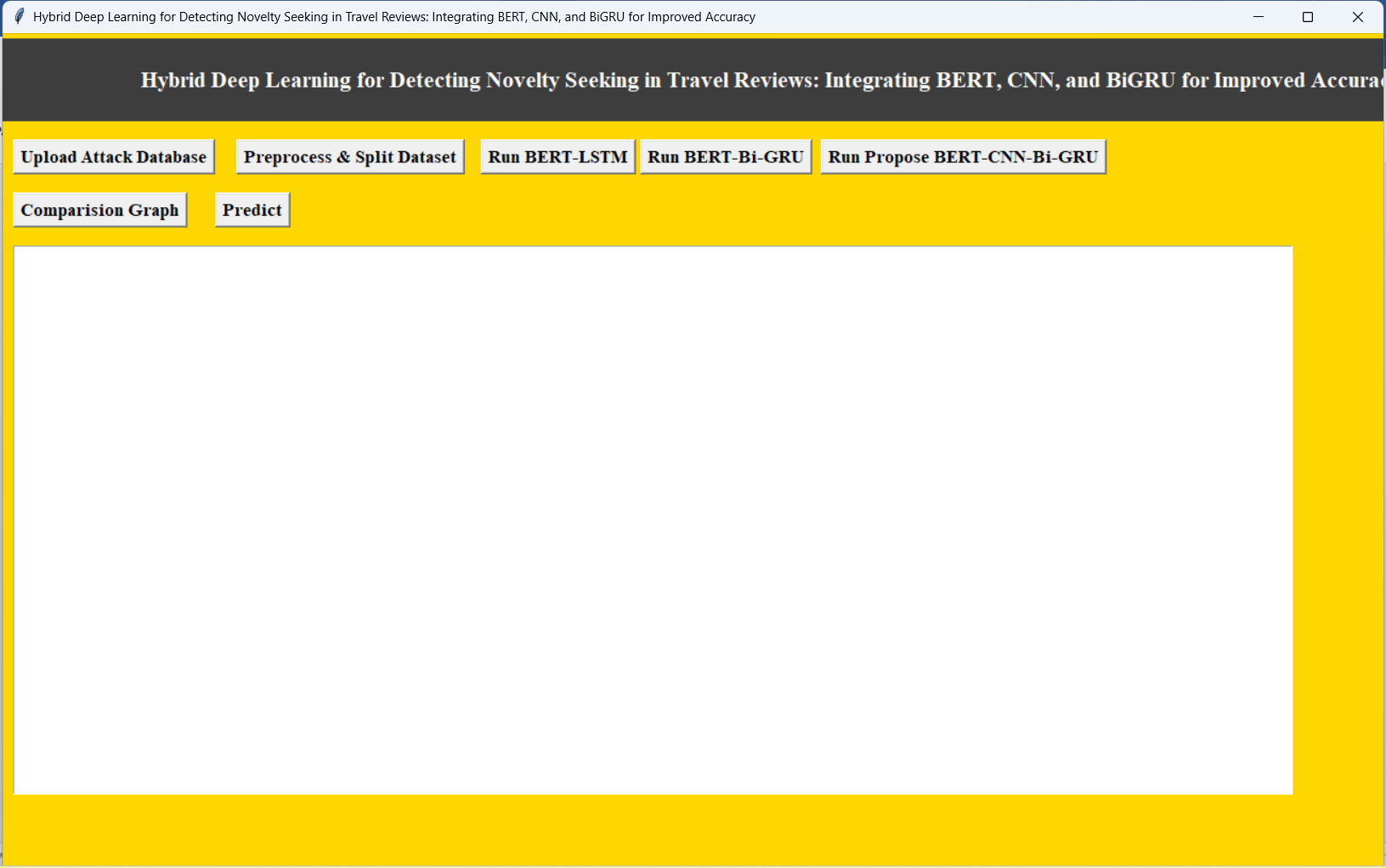
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

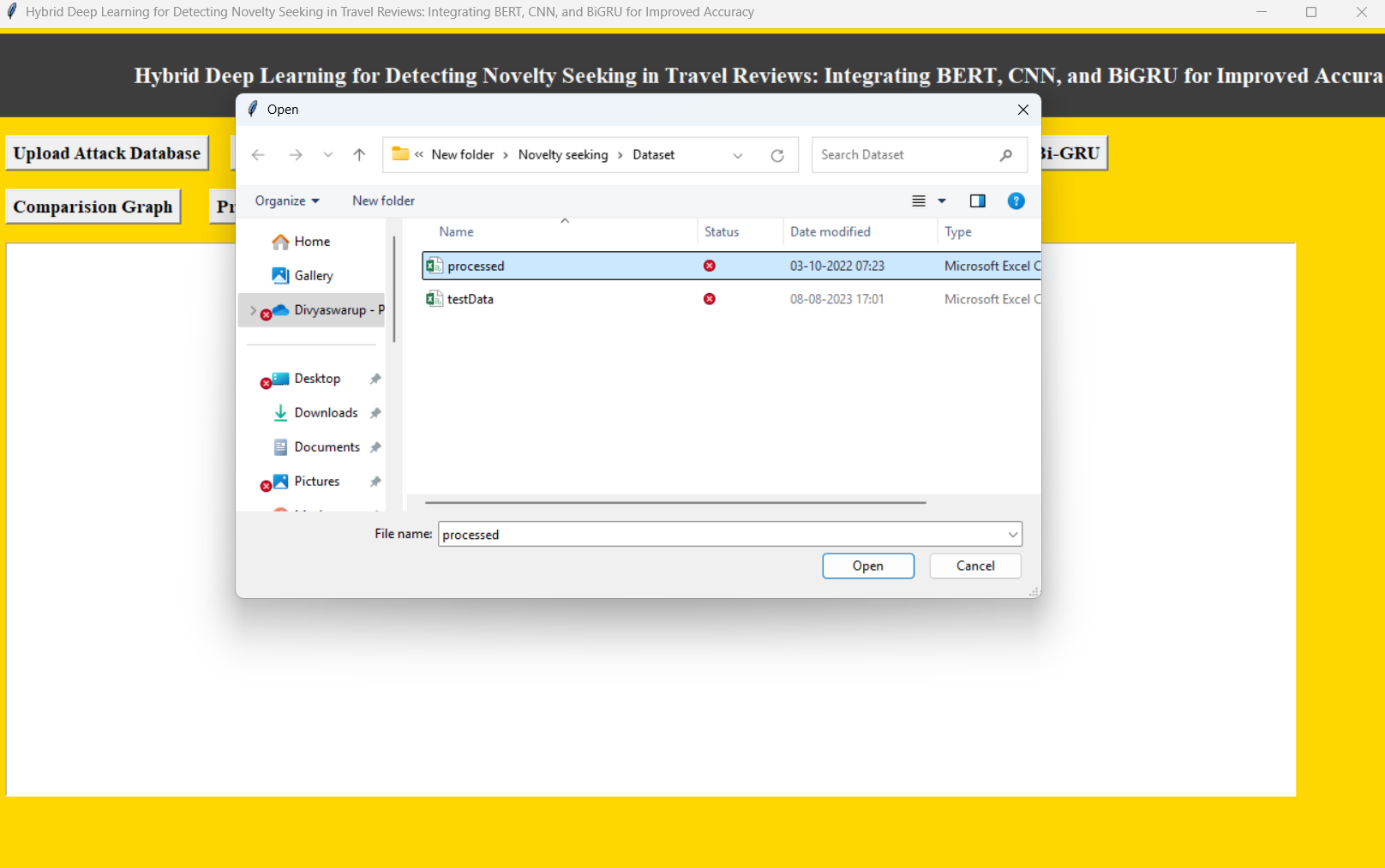
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload Attack Database | Verify Upload Attack Database Done or not | If upload Attack Database May not uploaded | we cannot do any further operations | we can do further operations | High | High |
| 02 | Preprocess & Split Dataset | Verify Preprocess & Split Dataset Done or not | If Preprocess & Split Dataset May not uploaded | we cannot do any further operations | we can do further operations | High | High |
| 03 | train BERT-LSTM algorithm | Verify train BERT-LSTM algorithm  submitted or not | If train BERT-LSTM algorithm  is not submitted | we cannot do any further operations | we can do further operations | High | High |
| 04 | train BERT-Bi-GRU algorithm | Verify  train BERT-Bi-GRU algorithm trained  or not | If train BERT-Bi-GRU algorithm  not obtained | we cannot do any further operations | we can do further operations | High | High |
| 05 | train BERT-CNN-Bi-GRU algorithm | Verify  train BERT-CNN-Bi-GRU algorithm trained  or not | If train BERT-CNN-Bi-GRU algorithm  not obtained | we cannot do any further operations | we can do further operations | High | High |
| 06 | performance graph | Verify performance graph trained or not | If performance graph not trained | we cannot do any further operations | we can do further operations | High | High |
| 07 | Predict | Verify Predict  submitted or not | If Predict  is not submitted | we cannot do any further operations | we can do further operations | High | High |

**7. SCREENSHOTS:**

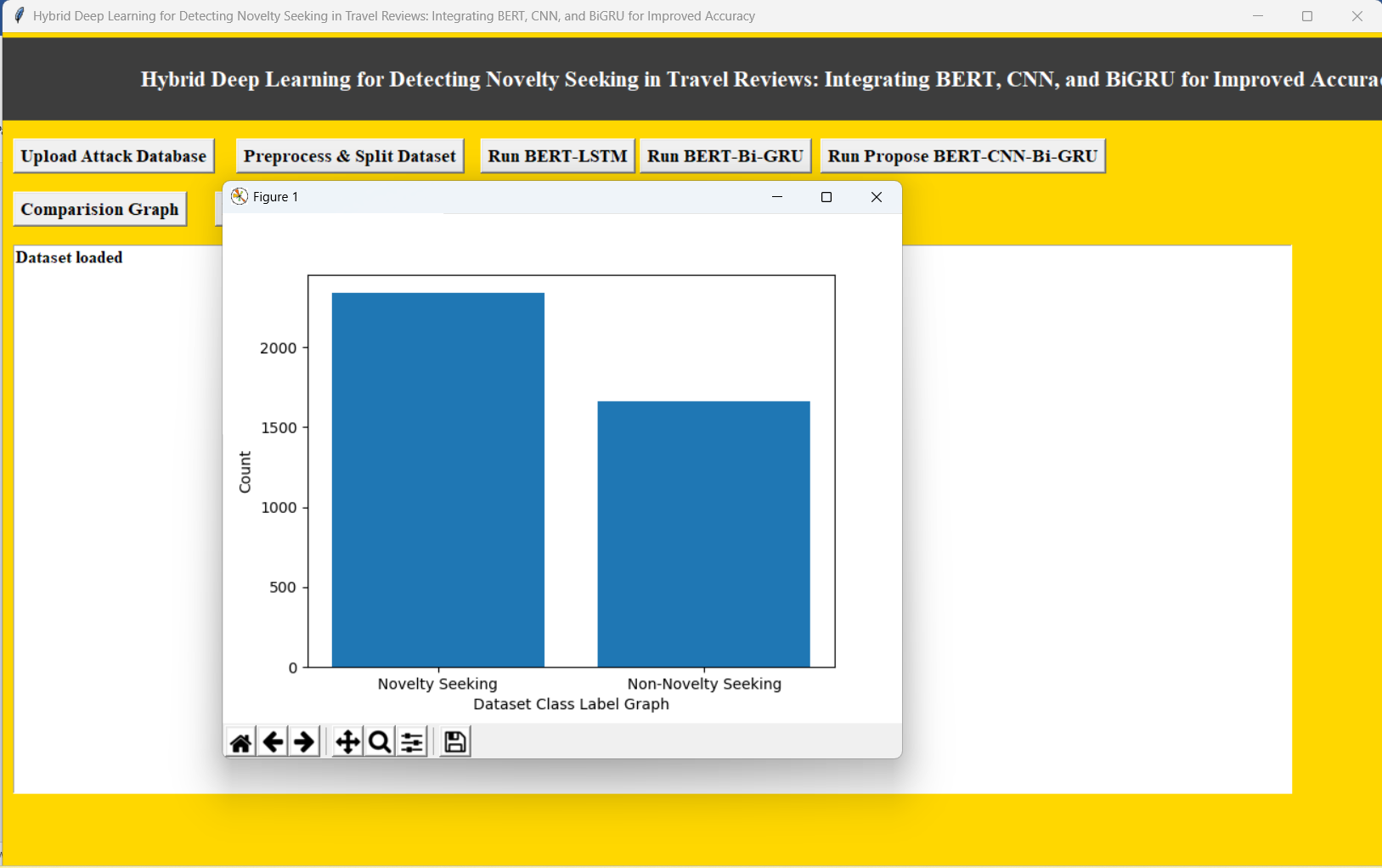
To Run the application, Click on “run.bat” file from the file location.

****

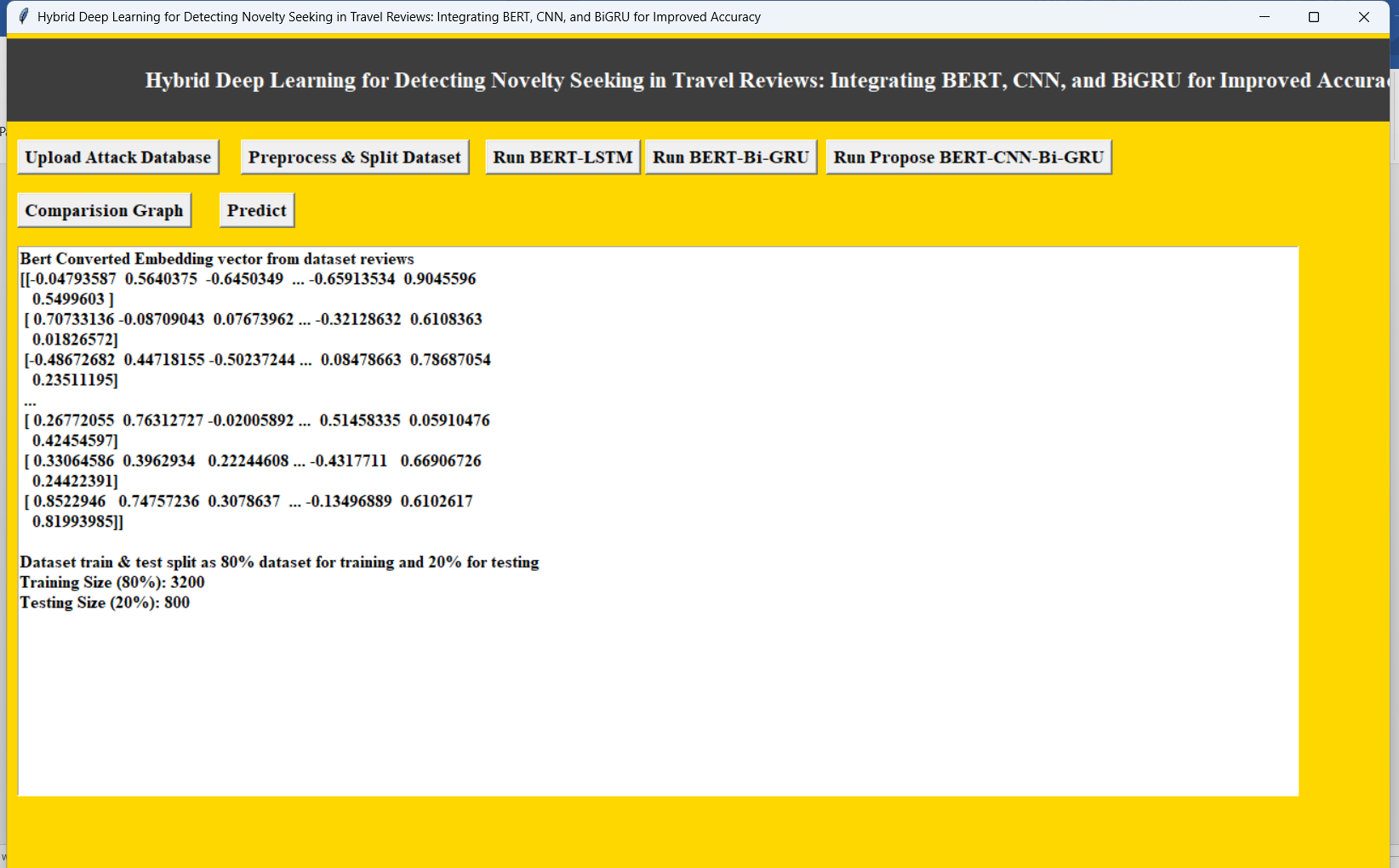
In Above screen select Upload Attack Database

****

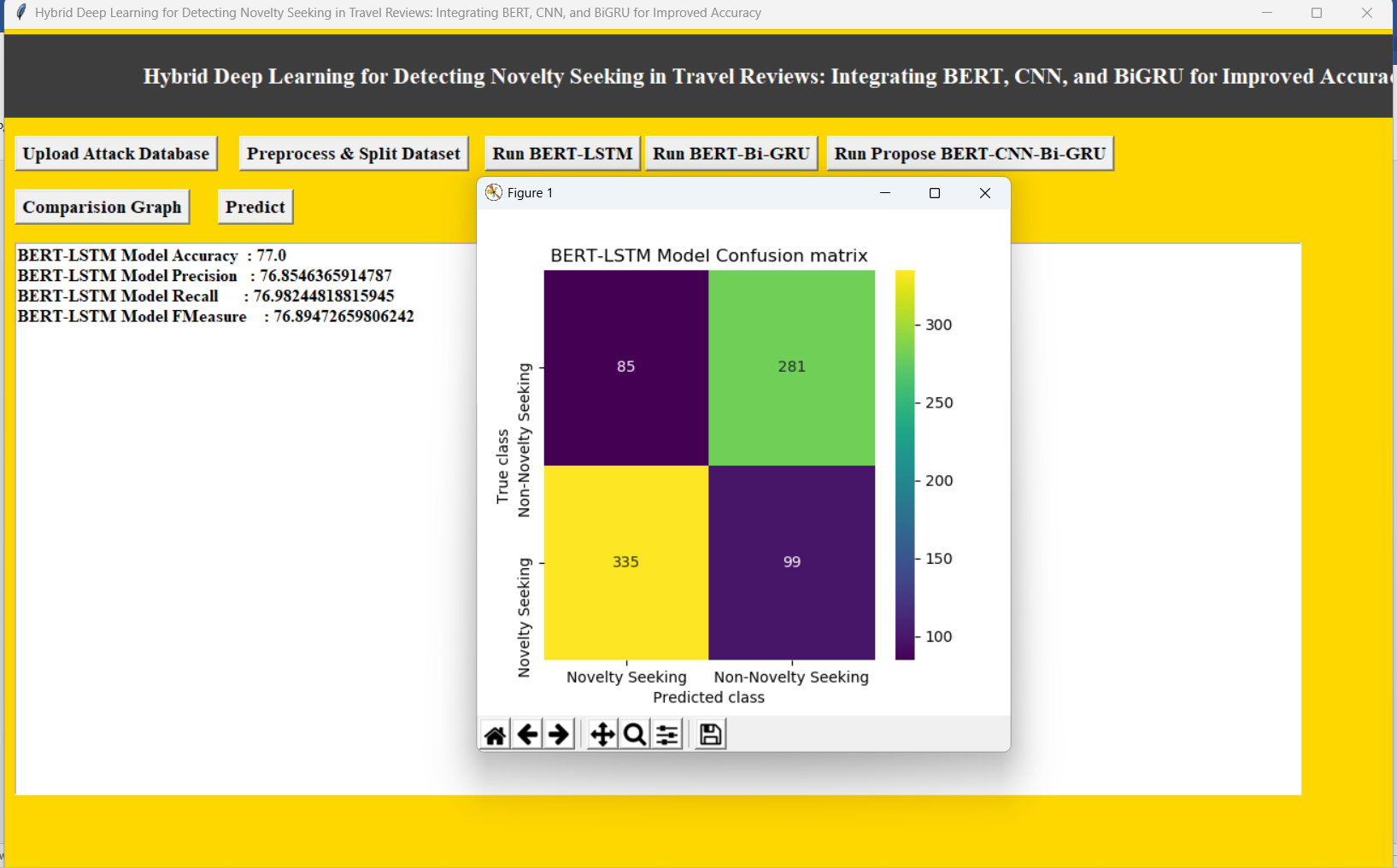
In above screen uploading processed dataset

****

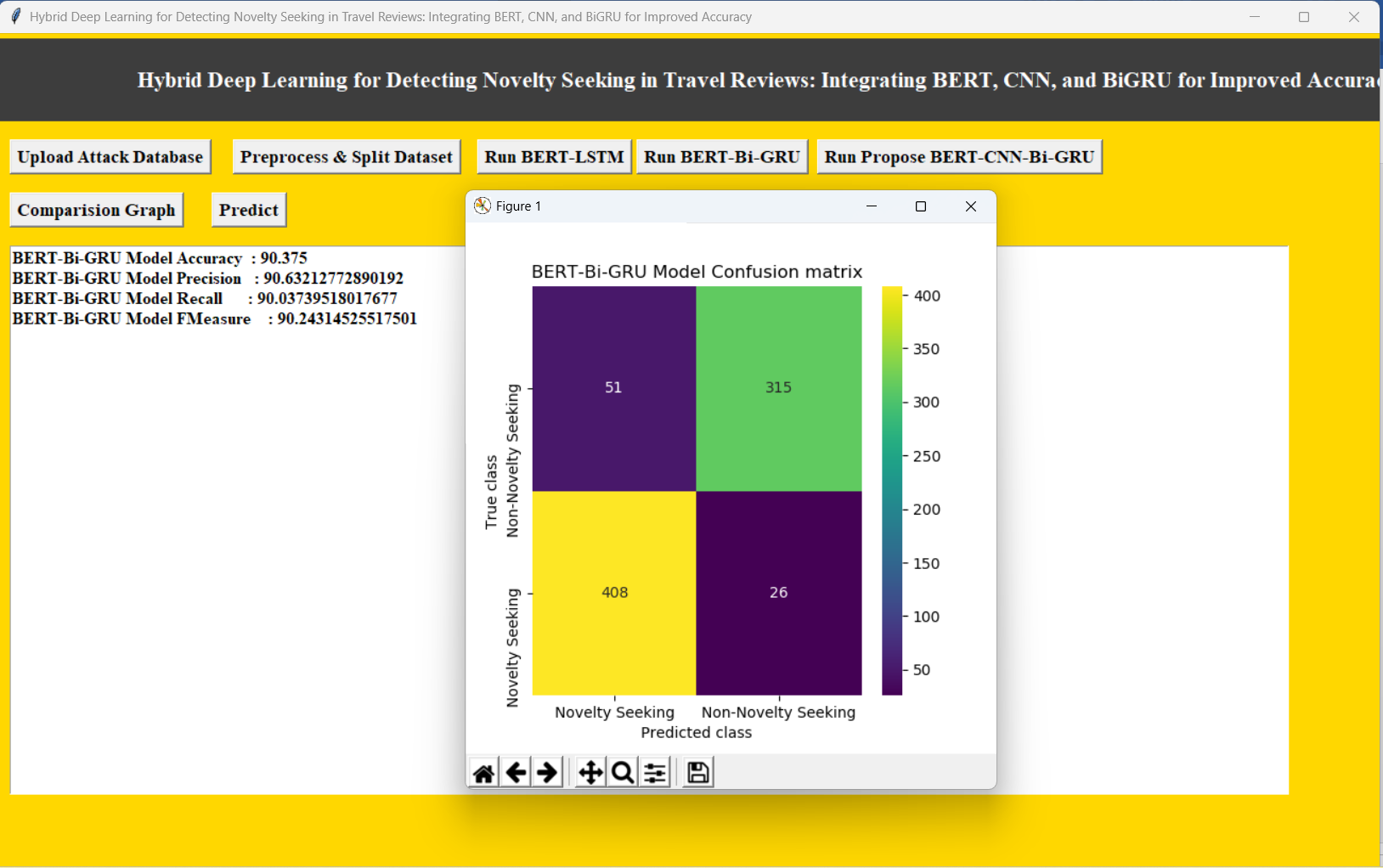
In above screen finding and plotting graph of Novelty and Non-Novelty reviews where x-axis represents novelty type and y-axis represents count

****

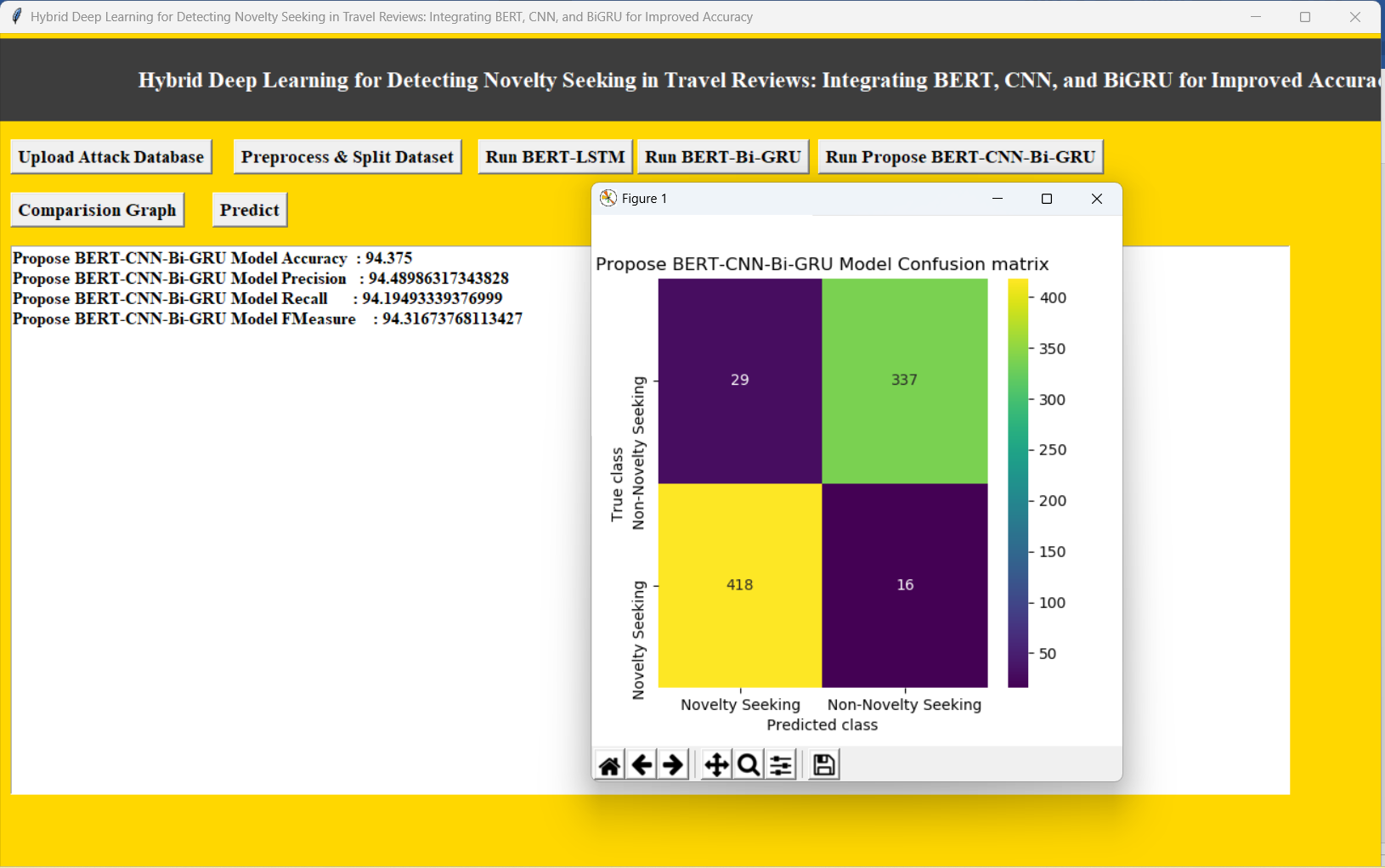
In above screen processing dataset such as shuffling and splitting into train and test and then showing output of train and test data and then defining arrays to store accuracy and other metrics values

****

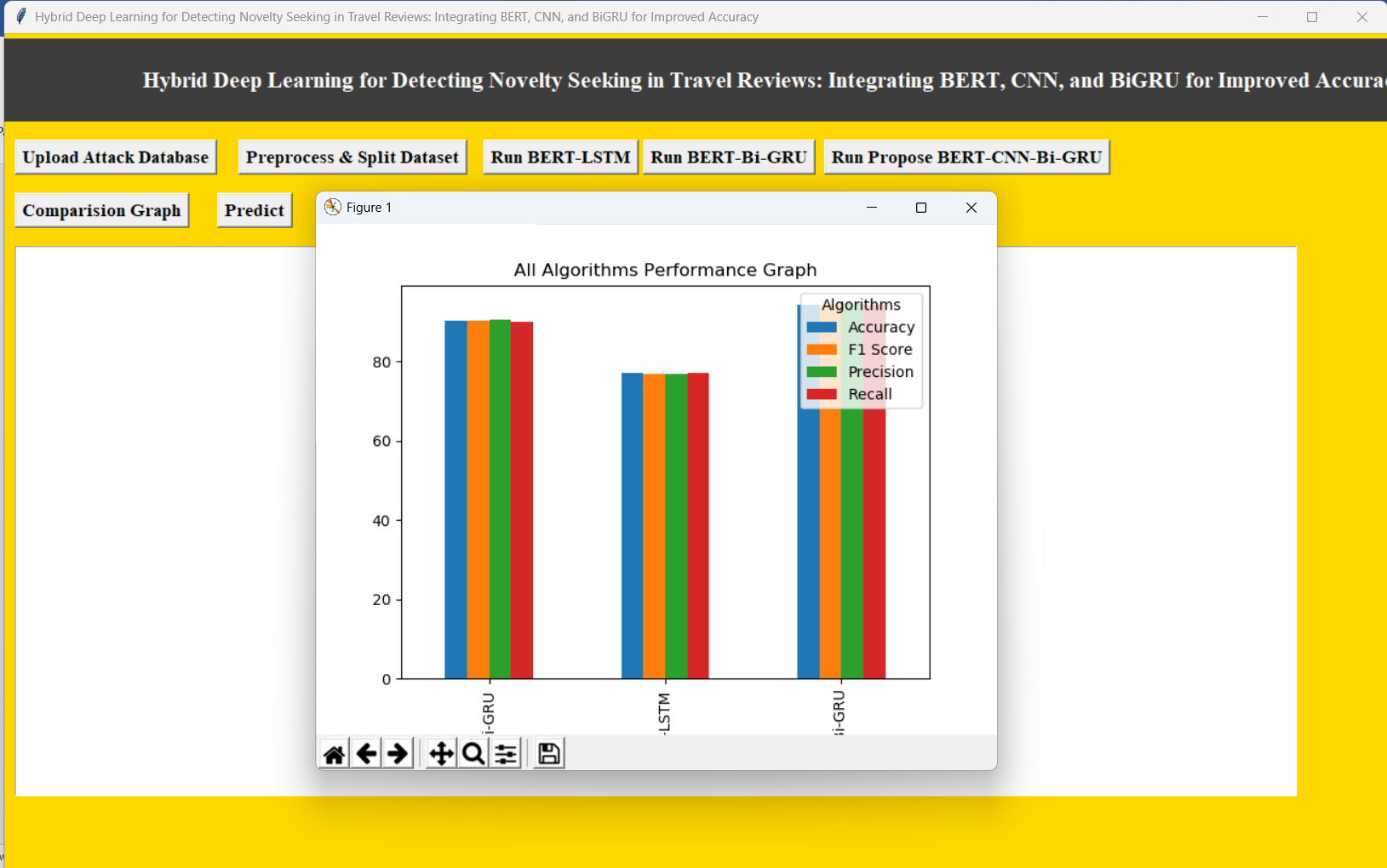
In above screen BERT-LSTM got 77% accuracy and can see other metrics also and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels where different colour boxes contains correct prediction count and all blue boxes contains incorrect prediction count which are very few.

****

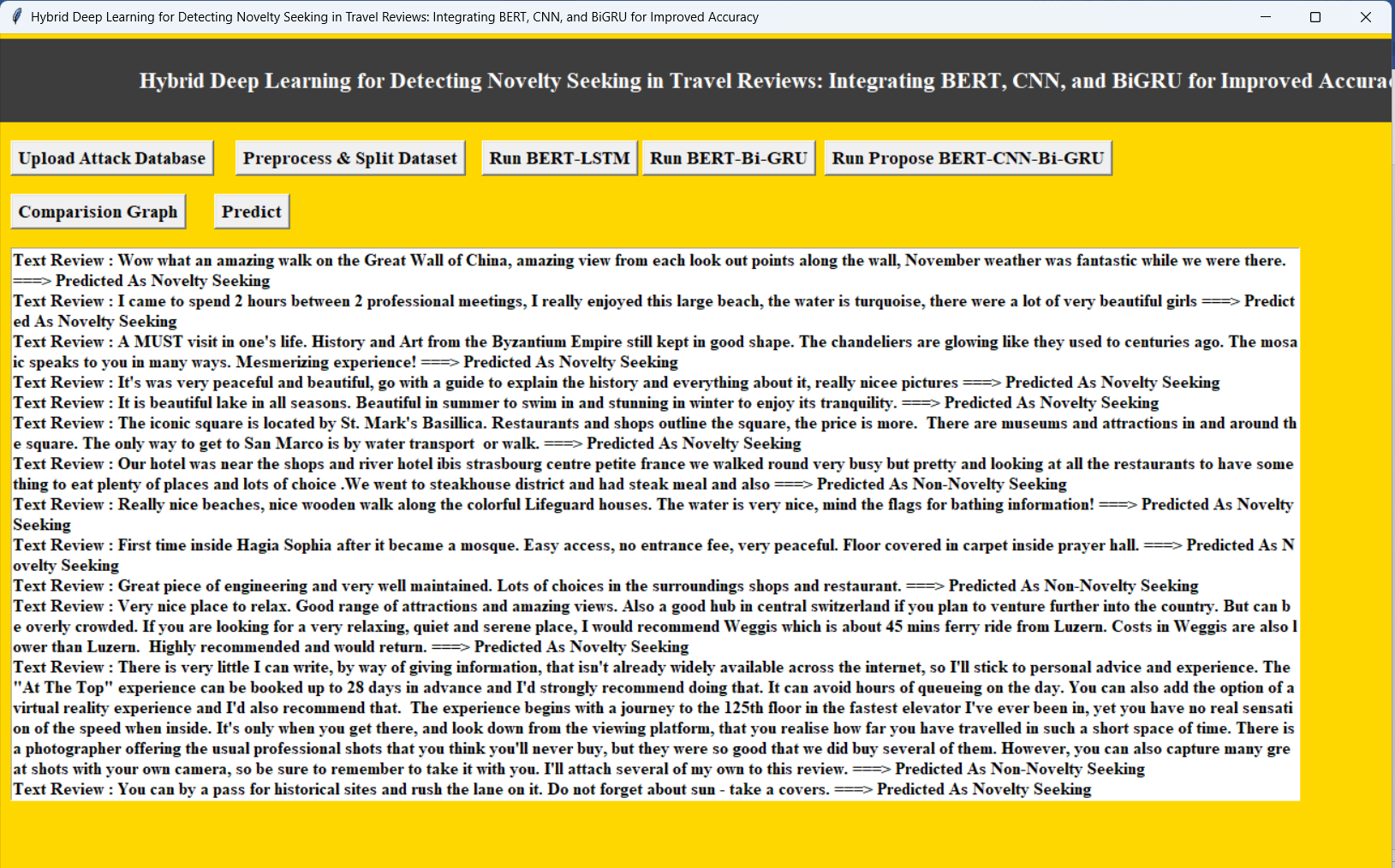
In above screen BERT-BI-GRU got 90% accuracy

****

In above screen Propose BERT-CNN-Bi-GRU model got 94% accuracy

****

In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms, propose got high performance

****

In above screen reading TEST reviews and then converting to BERT features and then using Propose model predicting Novelty Seeking or not and in output before arrow symbol we can see TEST data and after arrow symbol =🡺 we can see predicted output as Novelty Seeking or Non-Novelty seeking.

**8. CONCLUSION:**

This research demonstrates the applicability of deep learning in automatically processing substantial amounts of travel online reviews using a theory-based classification of NS personality traits. It proves that personality traits can be effectively and automatically identified based on the advanced computational techniques. However, there are some limitations to this research. NS dimensions are subjective, and the concept involves many dimensions with a high degree of abstraction. In addition to using a simple two-category for novelty recognition, multi-category recognition can also be performed based on the number of dimensions of the scale. Optimization of tourist destination recommendation system based on NS personality can be a further research direction, such as grouping users with NS, improving user portrait understanding and precision marketing.

**9. REFERENCES:**

[1] Y. Chang, R.-J. Hou, K. Wang, A. P. Cui, and C.-B. Zhang, "Effects of intrinsic and extrinsic motivation on social loafing in online travel communities," Computers in Human Behavior, vol. 109, p. 106360, 2020.

[2] G. Assaker, "Age and gender differences in online travel reviews and user-generated-content (UGC) adoption: extending the technology acceptance model (TAM) with credibility theory," Journal of Hospitality Marketing & Management, vol. 29, no. 4, pp. 428-449, 2020/05/18 2020, doi: 10.1080/19368623.2019.1653807.

[3] M. J. García, J. R. Chico, A. R. P. Sánchez, and J. A. L. Sánchez, "Does Electronic Word-of-mouth Differently Influence Traditional Economy Vs. Collaborative Economy Tourist Accommodation? An Empirical Study," Journal of Quality Assurance in Hospitality & Tourism, pp. 1-25, 2021, doi: 10.1080/1528008X.2021.1913692.

[4] I. Šagovnović and S. Kovačić, "Predicting Travel Motivation with Personality and Personal Values–The Roles of Big Five Plus Honesty/Humility Personality Traits and Kahle’s Values," International Journal of Hospitality & Tourism Administration, pp. 1-30, 2022.

[5] D. J. Yoon, "Rude customers and service performance: Roles of motivation and personality," The Service Industries Journal, vol. 42, no. 1-2, pp. 81-106, 2022.

[6] A. Goreis and M. Voracek, "A systematic review and metaanalysis of psychological research on conspiracy beliefs: Field characteristics, measurement instruments, and associations with personality traits," Frontiers in psychology, vol. 10, p. 205, 2019.

[7] J. Anglim and P. O’connor, "Measurement and research using the Big Five, HEXACO, and narrow traits: A primer for researchers and practitioners," Australian Journal of Psychology, vol. 71, no. 1, pp. 16-25, 2019.

[8] Y. Mehta, N. Majumder, A. Gelbukh, and E. Cambria, "Recent trends in deep learning based personality detection," Artificial Intelligence Review, vol. 53, no. 4, pp. 2313-2339, 2020.

[9] E. C. Hirschman, "Innovativeness, novelty seeking, and consumer creativity," Journal of consumer research, vol. 7, no. 3, pp. 283-295, 1980.

[10] S. S. Jang and R. Feng, "Temporal destination revisit intention: The effects of novelty seeking and satisfaction," Tourism management, vol. 28, no. 2, pp. 580-590, 2007.

[11] G. Assaker and R. Hallak, "Moderating effects of tourists’ novelty-seeking tendencies on destination image, visitor satisfaction, and short-and long-term revisit intentions," Journal of Travel Research, vol. 52, no. 5, pp. 600-613, 2013.

[12] A. Lloyd, R. McKay, C. L. Sebastian, and J. H. Balsters, "Are adolescents more optimal decision-makers in novel environments? Examining the benefits of heightened exploration in a patch foraging paradigm," Developmental Science, vol. 24, no. 4, p. e13075, 2021.

[13] E. E. Pujiastuti, "Novelty Seeking To Predict Behavior Intention in Rural Destination," Jurnal Bisnis dan Manajemen, vol. 7, no. 2, 2020.

[14] W.-J. Seo, L. A. Lewin, B. Moon, and B.-Y. Moon, "Moderating effects of affective image on relationship between noveltyseeking and behavioural intentions of runners," South African Journal for Research in Sport, Physical Education and Recreation, vol. 40, no. 3, pp. 105-120, 2018.

[15] Q. Nguyen, H. Nguyen, and T. Le, "Relationships among novelty seeking, satisfaction, return intention, and willingness to recommend of foreign tourists in Vietnam," Management Science Letters, vol. 10, no. 10, pp. 2249-2258, 2020.

[16] L.-H. Lee, Y. Lu, P.-H. Chen, P.-L. Lee, and K.-K. Shyu, "NCUEE at MEDIQA 2019: medical text inference using ensemble BERT-BiLSTM-Attention model," in Proceedings of the 18th BioNLP Workshop and Shared Task, 2019, pp. 528-532.

[17] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," Information, vol. 10, no. 4, p. 150, 2019.

[18] A. Rantanen, J. Salminen, F. Ginter, and B. J. Jansen, "Classifying online corporate reputation with machine learning: a study in the banking domain," Internet Research, vol. 30, no. 1, pp. 45-66, 2019, doi: 10.1108/intr-07-2018-0318.

[19] S. Li, M. You, D. Li, and J. Liu, "Identifying coal mine safety production risk factors by employing text mining and Bayesian network techniques," Process safety and environmental protection, vol. 162, pp. 1067-1081, 2022.

[20] C. Zhang, T. Tong, and Y. Bu, "Examining differences among book reviews from various online platforms," Online Information Review, 2019.