International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Neural Network-driven Duplicate Question Detection in CQA Systems

Ms. B. Shanthi , P. Raghumithra Reddy , K. Jagapathi , P. Ajay
CMR Institute of Technology

INTRODUCTION

1.1 PROBLEM STATEMENT

Community-based question answering platforms depend on the quality and uniqueness of user-submitted content. However, as more users join and contribute, the number of duplicate questions increases significantly. Users often post the same query with different wording or context, making it difficult for others to find existing solutions. This leads to repeated discussions, delays in receiving accurate answers, and an overall decline in the usability of the platform.

Traditional duplicate detection methods rely primarily on manual review or shallow text similarity rules. These approaches struggle to capture deeper semantic meaning, especially when questions look different on the surface but express the same intention. The manual approach is time-consuming, inconsistent across moderators, and not scalable to the millions of questions posted on such platforms.

There is also a lack of integrated systems that combine modern deep learning architecture with effective semantic embeddings to detect duplicates with high accuracy. Without such systems, duplicate questions continue to accumulate, reducing the efficiency of search tools and burdening both moderators and contributors. This project aims to address this gap by building a deep learning-based duplicate question detection system that understands context, meaning, and structure of questions beyond their surface-level similarities.

1.2 OBJECTIVE OF THE PROJECT

1.2.1 PRIMARY OBJECTIVES

- **To develop an AI-driven system for automated duplicate question detection**
    - Build reliable deep learning models that analyse question pairs and classify whether they are duplicates with high accuracy.
- **To implement Word2Vec for semantic feature representation**
    - Generate meaningful word embeddings that capture contextual and semantic relationships between words beyond simple text matching.
- **To design and compare deep learning architectures (CNN, RNN, LSTM)**
    - Evaluate multiple neural models to identify the architecture that best captures semantic similarity between questions.
- **To integrate effective text preprocessing for improved model performance**
    - Apply cleaning, tokenization, stopword removal, and vectorization to enhance the quality of input data and strengthen feature extraction.
- **To develop a user-friendly interface for real-time similarity prediction**
    - Provide users, moderators, and developers with a simple platform to input questions and instantly view duplicate detection results.
- **To improve platform efficiency and knowledge organization**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

- Enhance search effectiveness and reduce redundant content by automating the detection of repeated or closely related questions.

1.2.2 SECONDARY OBJECTIVES

- **To reduce manual moderation effort and avoid subjective judgment**
  - Automate duplicate identification to minimize human workload and eliminate inconsistencies caused by manual review.
- **To support large-scale community platforms through intelligent content management**
  - Help platforms like Stack Overflow, Quora, and similar CQA systems maintain cleaner threads and a more structured knowledge base.

1.3 SCOPE OF THE PROJECT

The scope of this project focuses on the development and implementation of an intelligent duplicate question detection system for Community-Based Question Answering (CQA) platforms. The system is designed to analyze pairs of user-generated questions and determine whether they represent the same underlying query, even when expressed using different wording or contextual variations. By leveraging advanced deep learning techniques, the project aims to address the limitations of traditional rule-based systems and provide a more accurate and scalable solution for managing redundant content.

The project encompasses the complete pipeline of duplicate detection, starting from data collection and preprocessing to model training, evaluation, and deployment. It includes the use of text preprocessing techniques such as cleaning, normalization, tokenization, and stop-word removal to prepare the data for analysis. The system further incorporates Word2Vec-based word embeddings to convert textual data into meaningful numerical representations that capture semantic relationships between words. These embeddings serve as input to deep learning models, enabling the system to understand contextual similarities between questions.

Within its scope, the project explores and compares multiple deep learning architectures, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks. Each of these models contributes uniquely to the detection process by capturing local patterns, sequential dependencies, and long-range contextual information in text. The project evaluates the performance of these models using standard metrics such as accuracy, precision, recall, and F1-score to determine the most effective approach for duplicate detection.

The system is also designed to provide real-time predictions, allowing users to input a query and receive immediate feedback on whether similar or duplicate questions already exist. This makes the system suitable for integration into live platforms where instant response is critical. Additionally, the project supports scalability by enabling the processing of large datasets, making it applicable to high-traffic platforms with millions of user queries.

However, the scope of the project is primarily limited to text-based duplicate detection using supervised learning techniques. It does not include advanced transformer-based architectures such as BERT or GPT, multimedia data processing (such as images or videos), or fully optimized large-scale deployment in production environments. These areas can be considered for future enhancements.

Overall, the scope of the project lies in developing a reliable, efficient, and scalable deep learning-based solution for detecting duplicate questions, improving content organization, and enhancing the overall user experience in CQA platforms.

2. LITERATURE REVIEW

2.1 EXISTING SYSTEM

The existing system used in Community-Based Question Answering (CQA) platforms relies on a traditional, rule-based duplicate question detection approach. Instead of using advanced machine learning or deep learning models, this system depends on five manually engineered similarity features to compare two questions and determine whether they express the same meaning. These features strictly focus on analyzing both lexical overlap and semantic relationships within the text.

The first feature, **cosine similarity**, measures how close two questions are in a vector space by comparing the frequency of their terms. It helps identify questions that share a similar vocabulary or structure, making it useful for detecting duplicates with

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

similar word patterns. The second feature, **term overlap**, calculates the number of common words between two questions. A high overlap indicates that the questions might be duplicates, although it may not capture deeper semantic relationships.

The third feature, **entity overlap**, extracts important entities—such as names, technical terms, concepts, or keywords—from the text and compares them across questions. If two questions refer to the same key entities, they are more likely to represent the same problem. The fourth feature, **entity type overlap**, goes one level deeper by comparing the categories or types of the extracted entities. Even if exact entities differ, matching entity types (e.g., programming languages, APIs, locations, tools, or topics) suggest that the questions belong to the same context or domain. This helps in detecting duplicates that use different words but refer to the same type of concept.

The fifth and final feature is **WordNet similarity**, which uses a linguistic database to analyze semantic relationships between words. This feature allows the system to detect duplicate questions even when different synonyms or phrases are used. For example, “error” and “bug” may be recognized as semantically related, helping the system identify paraphrased duplicates.

Overall, the existing system evaluates these five features separately and then combines their scores to determine whether two questions should be classified as duplicates. While effective to some extent, this approach has several limitations. It struggles with complex sentence structures, paraphrased questions, context-dependent meaning, and informal language commonly found in CQA communities. Since it relies solely on handcrafted features and lacks contextual understanding, the system cannot fully capture deeper semantic relationships that modern neural models can. This creates a need for more advanced machine learning and deep learning–based methods that can better interpret meaning, context, and variations in user-generated text.

2.2 LITERATURE SURVEY

The problem of duplicate question detection in Community-Based Question Answering (CQA) platforms has been extensively studied by researchers, leading to the development of various approaches ranging from traditional machine learning techniques to advanced deep learning models. This section reviews some of the significant contributions in this domain and highlights their methodologies, findings, and limitations.

Correa and Sureka (2014) – Analysis of Question Deletion in CQA Platforms

Correa and Sureka (2014) conducted a comprehensive study on question deletion patterns in CQA platforms to understand the factors influencing the removal of low-quality or redundant questions. Their research analyzed historical data and identified key trends such as increasing deletion rates, user-initiated deletions, and rapid moderator intervention. They also developed a predictive model using linguistic, syntactic, and user-based features to estimate the likelihood of a question being deleted at the time of posting. Their work emphasized the importance of automated systems for maintaining content quality and reducing redundancy in large-scale platforms.

Zhang et al. (2015) – Multi-Factor Duplicate Question Detection using DupPredictor

Zhang et al. (2015) proposed a multi-factor approach for duplicate question detection, known as DupPredictor. Their method combined multiple similarity measures, including title similarity, description similarity, topic similarity, and tag similarity, to provide a comprehensive evaluation of duplication. By incorporating Latent Dirichlet Allocation (LDA), they were able to capture latent semantic topics beyond surface-level text analysis. Experimental results demonstrated that integrating multiple semantic and textual features significantly improves detection accuracy. However, the approach still relied on handcrafted features and required careful feature engineering.

Silva et al. (2018) – Reproducibility Study of Duplicate Detection Models

Silva et al. (2018) focused on the reproducibility and validation of existing duplicate detection models. Their study highlighted the challenges associated with inconsistent datasets, imbalanced data distribution, and variations in evaluation methodologies. By re-evaluating several previously proposed models across different datasets, they observed significant performance variations, indicating that many models lacked generalization capability. Their findings stressed the need for standardized evaluation practices and more robust semantic modeling techniques.

Kamath et al. (2018) – Comparison of Machine Learning and Deep Learning Approaches

Kamath et al. (2018) conducted a comparative study between traditional machine learning algorithms and deep learning models for text classification tasks. Their research demonstrated that deep learning approaches, such as Convolutional Neural

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Networks (CNN) and Long Short-Term Memory (LSTM) networks, outperform classical methods like Support Vector Machines (SVM) and Naïve Bayes, particularly in tasks requiring semantic understanding and contextual interpretation. The study concluded that deep learning models are more suitable for complex natural language processing tasks, including duplicate question detection.

Overall, the literature indicates a clear transition from rule-based and feature-engineered approaches to deep learning-based methods for duplicate question detection. While earlier techniques provided foundational insights, they were limited in capturing deep semantic relationships and handling linguistic variations. Recent advancements in neural networks and word embeddings have significantly improved the ability to understand contextual meaning, making them more effective for large-scale CQA platforms. These studies collectively highlight the need for scalable, accurate, and adaptive systems, which forms the basis for the proposed deep learning-based approach in this project.

2.3 LIMITATIONS OF EXISTING SYSTEM

1. Limited Understanding of Context and Meaning

The existing system relies only on the surface-level features such as term overlap and cosine similarity. These features fail to capture deep contextual meaning. For example, two sentences may use entirely different words but still express the same idea. Since the system does not understand semantics beyond basic WordNet relationships, it cannot properly detect paraphrased or contextually similar questions.

2. High Dependency on Exact Word Matching

Most features in the system depend heavily on the presence of identical or similar words in both questions. Questions that use synonyms, domain-specific paraphrases, or different phrasing patterns are often missed. This results in many false negatives, where truly duplicate questions remain undetected.

3. Inability to Handle Complex Linguistic Variations

CQA platforms contain informal language, abbreviations, technical jargon, misspellings, and diverse sentence structures.

Since the existing system does not incorporate linguistic models, it struggles to interpret:

- Long and complex questions
- Grammatically incorrect sentences
- Multi-sentence explanations
- Code-related or domain-specific terminology
  This leads to inaccurate similarity calculations.

4. No Learning Capability

The rule-based system cannot learn from new data or adapt to evolving user behavior. As the platform grows and language patterns change, the system always produces the same results because it does not update its internal knowledge. Unlike machine learning systems, it cannot improve accuracy over time.

5. Limited Semantic Awareness Despite WordNet Usage

Although WordNet similarity adds a basic level of semantic comparison, it is inadequate for modern, domain-specific contexts. WordNet does not cover technical terms, modern jargon, abbreviations, or newly emerging phrases commonly seen in CQA communities. This creates semantic blind spots for the system.

6. Scalability Issues in Large Platforms

The system evaluates multiple similarity features for each pair of questions, which becomes inefficient as the number of questions grows into millions. Due to its rule-based nature, it does not optimize large-scale matching, leading to slower duplicate detection in high-traffic environments.

7. High False Positives and False Negatives

Because the system is based on rigid rules, it often incorrectly marks:

- **Different questions as duplicates** (false positives)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

- **Actual duplicates as different** (false negatives)
  This affects user experience and contributes to clutter in the platform.

8. No Support for Deep Semantic Relationships or Intent Understanding

The system cannot identify:

- Similar questions with different structure
- Questions describing the same issue indirectly
- User intent behind the query
  This limits its ability to provide accurate duplicate matches and wastes community effort.

9. Weak Handling of Entity and Concept Variations

Although entity overlap and entity type overlap are included, they are simplistic and fail when:

- Questions mention different but related entities
- Entity extraction tools misidentify terms
- Names or concepts appear in ambiguous forms

This weakens the overall reliability of the duplicate detection mechanism.

2.4 ADVANTAGES OF PROPOSED SYSTEM

1. Captures Deep Semantic Meaning

Unlike the traditional rule-based system that depends on word matching, the proposed deep learning approach understands the underlying meaning of sentences. Word2Vec embeddings represent words based on their semantic relationships, allowing the system to identify duplicate questions even when they are phrased differently or use synonyms. This helps detect paraphrased duplicates that older systems often miss.

2. Handles Complex and Long Questions Effectively

Using RNN and LSTM architectures allows the system to model long-range dependencies and contextual relationships within a question. Many CQA posts contain multi-sentence descriptions, technical details, or step-by-step explanations. LSTMs preserve this information over long sequences, enabling the system to understand the overall context and accurately determine duplicate meaning.

3. Recognizes Important Phrases and Patterns

CNN layers extract the most informative phrases—such as error messages, problem descriptions, or technical keywords—regardless of where they appear in the sentence. This helps the system identify patterns like “how to fix”, “null pointer exception”, or “installation error”, which strongly indicate duplicate questions. Such local pattern recognition is not possible with basic similarity metrics.

4. Reduces Dependency on Handcrafted Features

The proposed system automatically learns feature representations during training, eliminating the need for manually engineered features like cosine similarity or term overlap. This reduces human effort, improves adaptability, and avoids the limitations of rigid rule-based approaches. The model continuously improves as more data becomes available.

5. More Robust to Linguistic Variations

CQA questions often contain:

- Typos
- Slang or informal expressions
- Technical terms
- Domain-specific abbreviations

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

- Variable sentence structures

Because Word2Vec and deep networks generalize across vocabulary and structure, the system remains robust against these variations. This leads to more reliable duplicate detection across multiple domains and user writing styles.

6. Learns and Adapts Over Time

The model continuously improves with more training data. As new types of questions appear, the system adjusts through retraining or fine-tuning. This learning ability is absent in traditional systems, which always rely on fixed rules and cannot adapt to unusual or newly emerging question patterns.

7. High Accuracy and Better Generalization

Deep learning models (CNN/RNN/LSTM) excel at pattern recognition and semantic understanding, enabling significantly higher accuracy in duplicate detection. The system generalizes well across unseen questions because it learns both global and local semantic features, reducing false positives and false negatives.

8. Effective for Large-Scale Platforms

CQA platforms contain millions of questions. Deep learning models can scale efficiently once trained, especially when combined with optimized inference techniques. The system can process large volumes of new questions quickly and identify duplicates with minimal computational overhead during inference.

9. Produces a Continuous Similarity Score

Instead of a binary decision, the system generates a similarity probability, which allows:

- Ranking of potential duplicates
- Better user recommendations
- Integration with search engines
- Higher recall for related questions

This continuous scoring ability makes the system more flexible and user-friendly.

10. Improves Community Experience

With more effective duplicate detection:

- Users find answers faster
- Redundant content is reduced
- Moderators spend less time managing duplicate questions
- The platform remains organized and efficient

Overall, the user experience becomes smoother, and the knowledge base becomes more structured.

2.5 FEASIBILITY STUDY

The feasibility study evaluates the practicality and viability of developing and implementing the proposed duplicate question detection system. It examines various aspects such as technical feasibility, economic feasibility, and operational feasibility to ensure that the system can be successfully developed, deployed, and maintained within the given constraints.

2.5.1 TECHNICAL FEASIBILITY

The proposed system is technically feasible as it is built using widely available and well-established technologies in the field of machine learning and natural language processing. The implementation utilizes programming languages such as Python along with powerful libraries and frameworks including TensorFlow, Keras, NumPy, Pandas, and Scikit-learn. These tools provide robust support for building, training, and evaluating deep learning models such as CNN, RNN, and LSTM.

The system requirements, including hardware specifications such as a standard processor, moderate RAM, and sufficient storage, are easily achievable in most modern computing environments. Additionally, the use of Word2Vec embeddings for semantic

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

representation and the integration of preprocessing techniques further enhance the technical capability of the system. Therefore, the project can be developed and executed without requiring specialized or high-cost infrastructure.

2.5.2 ECONOMIC FEASIBILITY

The project is economically feasible as it does not require significant financial investment. The development is carried out using open-source tools and libraries, which eliminates the need for expensive software licenses. Platforms such as Python and its associated frameworks are freely available, making the overall development cost minimal.

Moreover, the system can be deployed on existing infrastructure without requiring additional hardware investments. The maintenance cost is also low, as updates and improvements can be made using the same development environment. Thus, the project provides a cost-effective solution for duplicate question detection in CQA platforms.

2.5.3 OPERATIONAL FEASIBILITY

The proposed system is operationally feasible as it is designed to be user-friendly and easy to integrate into existing platforms. The interface allows users to input queries and receive duplicate detection results in real time, making it suitable for practical applications. The system reduces the need for manual moderation, thereby improving efficiency and consistency in managing content.

Furthermore, the system can be adapted to handle large datasets and can be scaled according to platform requirements. Its ability to provide accurate and timely results enhances user experience and supports effective knowledge management. With proper deployment, the system can operate smoothly in real-world environments with minimal user training.

Based on the analysis of technical, economic, and operational aspects, the proposed duplicate question detection system is highly feasible. It can be efficiently developed using available resources, implemented at a low cost, and successfully operated in real-world CQA platforms. The feasibility study confirms that the project is practical, sustainable, and capable of delivering significant improvements in content organization and user experience.

3. SYSTEM DESIGN

3.1 PROPOSED SOLUTION

The proposed system introduces an advanced deep learning–based framework for detecting duplicate questions in Community-Based Question Answering (CQA) platforms. Unlike traditional systems that rely on simple lexical overlap and rule-based similarity calculations, the proposed approach leverages powerful neural architectures—Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks—to understand the deeper semantic meaning of questions. To support this, Word2Vec is used to generate dense vector representations of words, enabling the system to capture complex linguistic patterns, contextual relationships, and hidden similarities that are not recognizable through basic text-matching techniques.

The system begins by preprocessing the textual content of each question. This includes normalizing the text, tokenizing it into words, and converting each word into a continuous-valued vector using Word2Vec. Unlike traditional bag-of-words models, these embeddings preserve semantic relationships—words with similar meanings are placed closer together in vector space. This allows

the system to detect duplicate questions even when they are phrased differently or use domain-specific synonyms.

Once embedded, questions are processed through deep learning encoders. CNNs are used to extract key local features and patterns, such as important phrases or repeated terms, which play a significant role in identifying duplicates. RNNs and LSTMs further enhance this capability by modeling the sequential nature of language. LSTMs, in particular, can capture long-term dependencies, contextual meaning, and relationships across sentences—important in CQA platforms where many questions contain multi-line descriptions and detailed explanations.

After processing through one or more deep learning models, each question is transformed into a fixed-length semantic representation. To determine whether two questions are duplicates, the system compares these representations using similarity operations such as concatenation, element-wise multiplication, and distance-based measures. These combined features are then passed through dense neural layers, which learn how to differentiate true duplicates from merely similar-looking but distinct questions. The final output is a probability score indicating the likelihood that the question pair represents a duplicate.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

This deep learning–based system significantly improves duplicate detection by capturing both surface-level text patterns and deeper semantic meaning. It reduces the reliance on handcrafted features, adapts to linguistic variations naturally, and provides far more accurate and consistent results compared to traditional rule-based approaches.

3.2 SYSTEM ARCHITECTURE

The system architecture of the proposed duplicate question detection model is divided into two major phases, namely the training phase and the prediction phase. This architecture is designed to identify duplicate questions in Community-Based Question Answering (CQA) platforms by combining text preprocessing, semantic embedding generation using Word2Vec, and deep learning-based similarity analysis. The overall flow ensures that historical question data is used to train the model, while newly submitted questions are processed in real time to retrieve the most relevant duplicate question pairs.

In the training phase, the process begins with the collection of historical questions from a dataset. These questions are first passed through a preprocessing module, where unnecessary symbols, noise, punctuation, and irrelevant text components are removed. The text is normalized and prepared in a structured format for further analysis. After preprocessing, the cleaned data is organized into historical question pairs. These pairs represent examples of questions that may either be duplicates or non-duplicates and serve as the training input for the learning system.

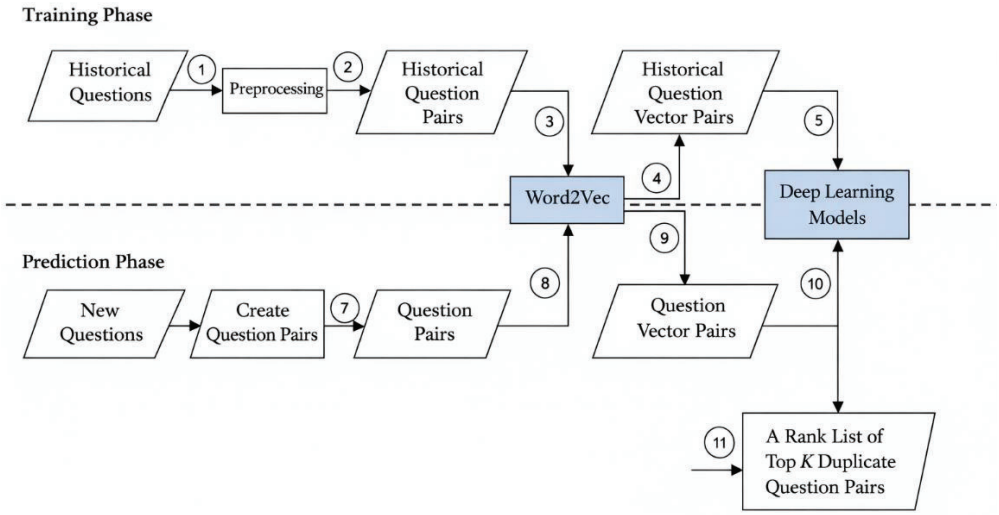

Figure 6.1: System Architecture

Next, the historical question pairs are passed to the Word2Vec module, which converts textual question pairs into dense numerical vector representations. Word2Vec plays a major role in the architecture by capturing semantic and contextual relationships between words. Instead of relying only on exact keyword matching, it transforms each question into a vector space where semantically similar words are positioned close to each other. After this transformation, the output becomes historical question vector pairs, which can be effectively processed by the learning models.

These vector pairs are then supplied to the deep learning models, which form the core of the duplicate detection system. In your project, these models include CNN, RNN, and LSTM, each contributing to semantic understanding in different ways. CNN helps in extracting local textual patterns and important phrase-level features, RNN captures sequential dependencies in the text, and LSTM improves the understanding of long-term contextual relationships in lengthy questions. Using the historical vector pairs, these models learn how to distinguish duplicate question pairs from non-duplicate ones. Thus, the training phase is responsible for building an intelligent model capable of understanding similarity beyond simple lexical overlap.

In the prediction phase, the architecture handles new questions entered by the user. When a new question is received, the system first creates possible question pairs by combining the new question with existing questions stored in the dataset. These generated

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

pairs are then sent to the Word2Vec module, where they are converted into question vector pairs using the same semantic embedding process applied during training. Maintaining the same embedding mechanism in both phases ensures consistency between training data and real-time input data.

The generated vector pairs are then passed to the already trained deep learning models, which analyze the semantic similarity between the new question and existing questions. Based on the learned patterns from the training phase, the model computes similarity scores and identifies the most relevant duplicate candidates. Finally, the system produces a ranked list of top K duplicate question pairs, where the most similar or likely duplicate questions are presented at the top. This ranking mechanism is highly useful because it not only identifies whether a duplicate exists but also provides the best matching alternatives to the user.

Thus, the proposed architecture supports both offline model training and online duplicate prediction in an effective and scalable manner. The training phase enables the system to learn meaningful semantic relationships from historical data, while the prediction phase allows real-time duplicate detection for new user queries. By integrating preprocessing, Word2Vec embeddings, and deep learning models into a single workflow, the architecture provides a robust solution for reducing redundancy, improving search efficiency, and enhancing knowledge organization in CQA platforms.

3.3 WORKFLOW OR BLOCK DIAGRAMS

3.3.1 CLASS DIAGRAM

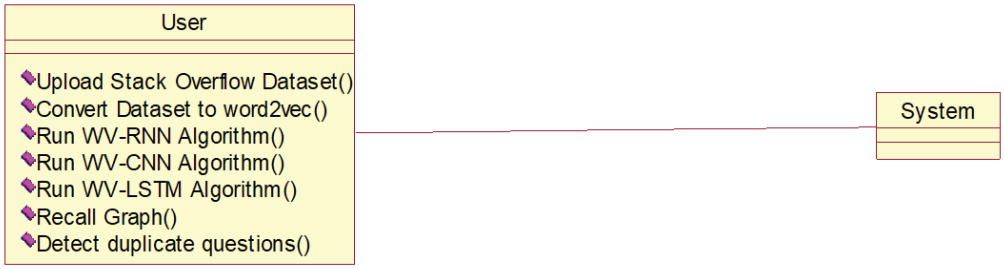

Figure 3.3.1: Class Diagram of the Proposed System

The class diagram represents the interaction between the **User** and the **System** in the duplicate question detection application. The **User** class acts as the primary interface through which all operations are initiated. It includes functionalities such as uploading the Stack Overflow dataset, converting the dataset into Word2Vec embeddings, executing deep learning algorithms (RNN, CNN, and LSTM), generating recall graphs, and detecting duplicate questions.

The **System** class processes the requests provided by the user and performs the corresponding operations internally. It handles data preprocessing, embedding generation, model execution, and result computation. The interaction between the User and System ensures a smooth workflow, where the user provides input and triggers actions, while the system performs backend processing and returns the output.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

3.3.2 DATAFLOW DIAGRAM

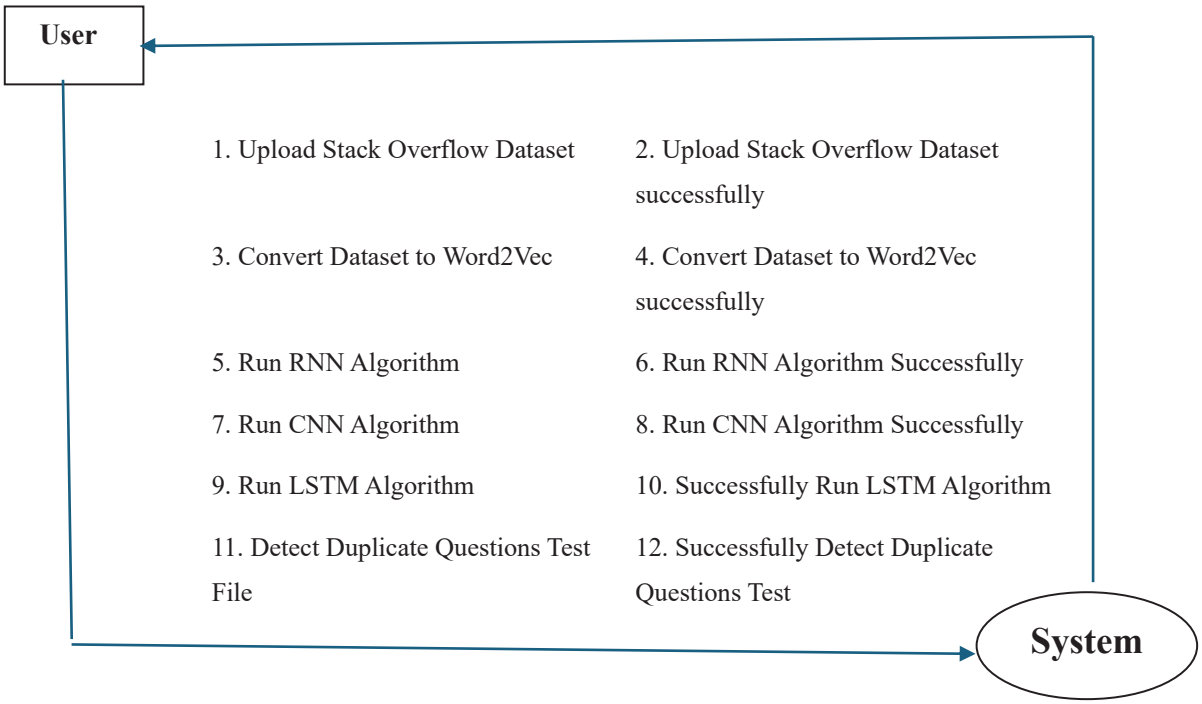

Figure 6.2.2: Dataflow Diagram

The data flow diagram shows how data moves through the duplicate question detection system. It begins with the user uploading the dataset, after which the system preprocesses the data, converts it into Word2Vec embeddings, applies deep learning algorithms such as CNN, RNN, and LSTM, and finally detects duplicate questions. The processed results are then provided back to the user.

3.3.3 USE CASE DIAGRAM

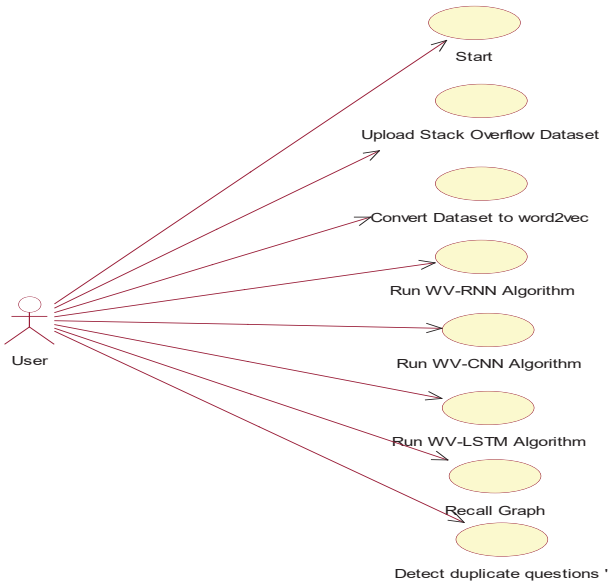

Figure 6.2.3: Use case Diagram

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

The use case diagram represents the interaction between the User and the duplicate question detection system. It shows that the user can perform operations such as uploading the dataset, converting data into Word2Vec embeddings, running CNN, RNN, and LSTM algorithms, and detecting duplicate questions. The system responds by processing these actions and generating the required results.

3.3.4 ACTIVITY DIAGRAM

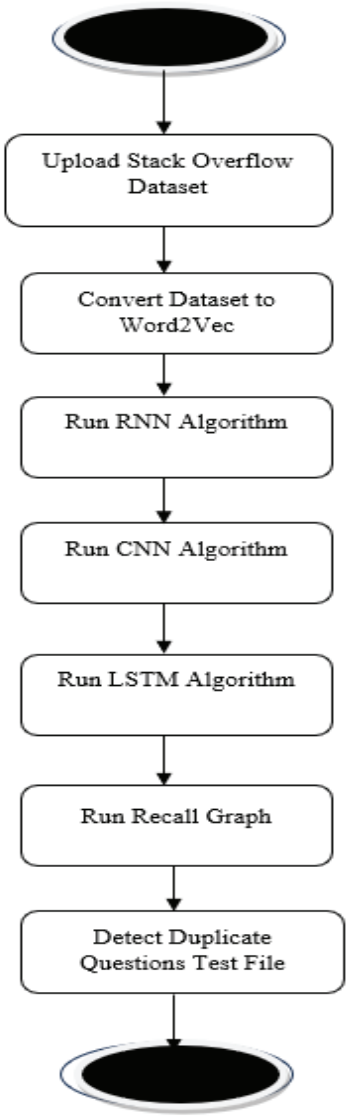

Figure 6.2.4: Activity Diagram

The activity diagram shows the step-by-step workflow of the duplicate question detection system. It starts with uploading the dataset, followed by preprocessing, Word2Vec conversion, execution of CNN, RNN, and LSTM models, and finally duplicate question detection. The process ends with displaying the output results to the user.

3.3.5 SEQUENCE DIAGRAM

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

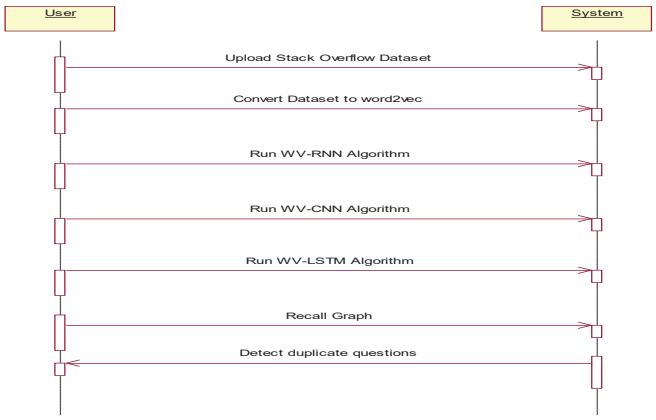

Figure 6.2.5: Sequence Diagram

The sequence diagram shows the order of interactions between the user and the system. It explains how the user uploads the dataset, initiates preprocessing and Word2Vec conversion, runs the CNN, RNN, and LSTM models, and finally receives the duplicate question detection results from the system.

6.2.6 COMPONENT DIAGRAM

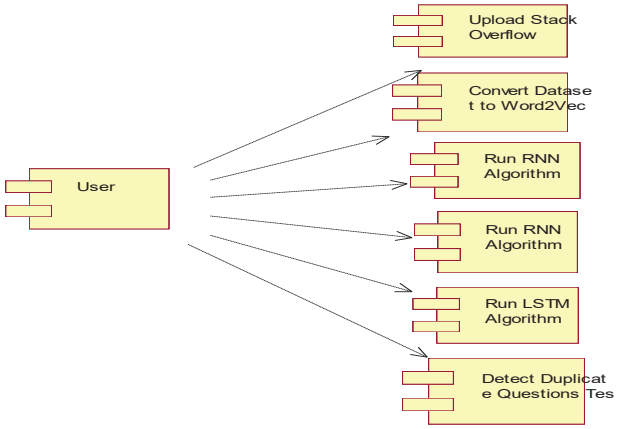

Figure 6.2.6: Component Diagram

The component diagram represents the main modules of the duplicate question detection system and their connections. It includes components such as dataset input, preprocessing module, Word2Vec embedding module, deep learning models (CNN, RNN, LSTM), and result output module, which together perform duplicate question detection efficiently.

3.3.7 DEPLOYMENT DIAGRAM

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

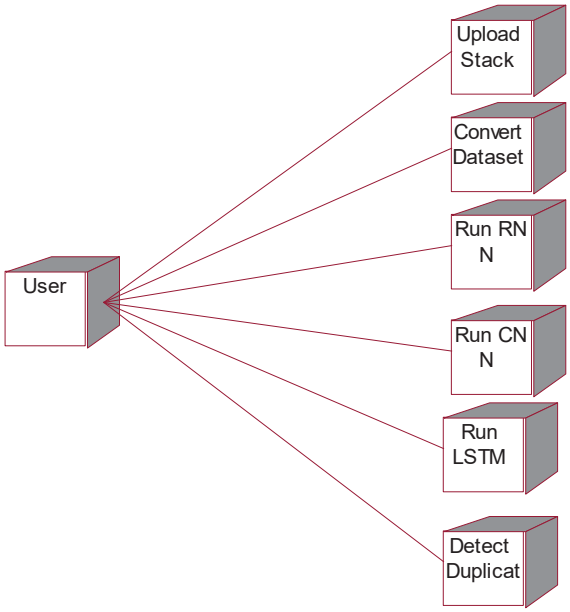

Figure 6.2.7: Deployment Diagram

The deployment diagram shows the physical arrangement of the duplicate question detection system. It represents how the user device interacts with the application system, where the dataset, preprocessing module, Word2Vec embedding module, deep learning models, and output generation are deployed to perform duplicate question detection and display the results.

3.4 DATA COLLECTION AND PREPROCESSING

Data collection and preprocessing form the foundational stages of the proposed duplicate question detection system. The quality and structure of the input data significantly influence the performance of the deep learning models used in this project. Therefore, careful attention is given to acquiring a reliable dataset and applying appropriate preprocessing techniques to ensure consistency, accuracy, and meaningful representation of textual information.

3.4.1 DATA COLLECTION

The dataset used in this project is obtained from Community-Based Question Answering (CQA) platforms such as Stack Overflow, which contain a large number of user-generated questions along with labeled information indicating whether a pair of questions is duplicate or not. Each data instance typically consists of two questions and a corresponding label, where the label indicates whether the questions represent the same underlying query.

The collected dataset includes diverse types of questions with variations in vocabulary, sentence structure, technical terminology, and writing style. This diversity is essential for training a robust model capable of handling real-world scenarios. The dataset is then divided into training, validation, and testing subsets. The training set is used to train the deep learning models, the validation set is used for tuning hyperparameters and preventing overfitting, and the testing set is used to evaluate the final performance of the system.

3.4.2 DATA PREPROCESSING

Raw textual data collected from CQA platforms often contains noise, inconsistencies, and irrelevant information that can negatively affect model performance. Therefore, preprocessing is applied to clean and standardize the data before feeding it into the model.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

The first step in preprocessing is text cleaning, where unwanted elements such as HTML tags, special characters, punctuation marks, and extra spaces are removed. This helps in reducing noise and improving the clarity of the input text. The text is then converted to lowercase to maintain uniformity and avoid duplication of words due to case differences.

The next step involves tokenization, where each question is split into individual words or tokens. Tokenization enables the model to process text at the word level and capture relationships between words. Following this, stop-word removal is performed to eliminate commonly used words such as “the,” “is,” and “and,” which do not contribute significantly to the meaning of the sentence.

To further enhance consistency, lemmatization or stemming techniques may be applied to reduce words to their base or root form. This ensures that different forms of the same word are treated uniformly. Additionally, handling of out-of-vocabulary (OOV) words is performed by replacing unknown or rare words with a special token, ensuring that the model can process unseen data effectively.

Since deep learning models require fixed-length input sequences, padding and truncation are applied to standardize the length of all question sequences. Short sequences are padded with zeros, while longer sequences are truncated to a predefined maximum length. This ensures uniform input dimensions across the dataset.

After preprocessing, the cleaned and structured text is ready for conversion into numerical representations using Word2Vec embeddings, which are then used as input to the deep learning models for duplicate question detection.

3.5 ALGORITHMS OF ML MODEL

The proposed duplicate question detection system utilizes multiple deep learning algorithms to effectively capture semantic similarity between question pairs. The primary models used in this project are Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks. Each of these algorithms contributes uniquely to understanding textual data and improving the accuracy of duplicate detection.

Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is used to extract local features and important patterns from text data. In this approach, the input question sequences are first converted into vector representations using Word2Vec embeddings. These embeddings are then passed through convolutional layers, where multiple filters slide over the input to capture meaningful n-gram features such as phrases and key patterns.

The convolution operation is followed by a pooling layer, typically max-pooling, which reduces the dimensionality of the feature maps and retains the most significant features. This process helps the model focus on the most relevant parts of the text. The extracted features are then flattened and passed through fully connected layers to produce the final representation. CNN is particularly effective in identifying repeated patterns, error messages, and important keywords that indicate duplicate questions.

Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) are designed to process sequential data by maintaining information about previous inputs in the sequence. In this model, each word in a question is processed one at a time, and the output of the previous step is used as input for the next step. This allows the network to capture the order and structure of words within a sentence.

RNNs are useful for understanding sentence flow and contextual dependencies between words. However, traditional RNNs face challenges such as the vanishing gradient problem, which limits their ability to capture long-range dependencies in lengthy sequences. Despite this limitation, RNNs provide a foundational approach for sequence modeling and contribute to capturing basic contextual relationships in the data.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are an advanced form of RNN designed to overcome the limitations of traditional RNNs. LSTM introduces memory cells and gating mechanisms, including input, forget, and output gates, which allow the model to retain important information over long sequences and discard irrelevant data.

In the context of duplicate question detection, LSTM is highly effective in capturing long-term dependencies and understanding the overall context of a question. This is particularly important for questions that contain multiple sentences or detailed descriptions. By preserving contextual information across the entire sequence, LSTM improves the model’s ability to identify semantically similar questions, even when they are phrased differently.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Overall Model Functioning

In the proposed system, the input question pairs are first converted into vector representations using Word2Vec embeddings. These embeddings are then processed through CNN, RNN, and LSTM models to extract different levels of semantic features. The outputs from these models are combined and passed through dense layers to compute a similarity score between the question pairs. Finally, a classification layer determines whether the questions are duplicates or non-duplicates.

3.6 TOOLS, TECHNOLOGIES AND SOFTWARE

The development of the duplicate question detection system involves the integration of multiple tools, technologies, and software frameworks that support data preprocessing, model training, system implementation, and deployment. Each component plays a crucial role in ensuring the accuracy, scalability, and performance of the system.

Programming Language

The system is primarily developed using Python, which provides extensive support for machine learning, natural language processing, and data analysis. Python's simplicity and rich ecosystem of libraries make it highly suitable for implementing deep learning-based solutions.

Machine Learning Framework

The model is implemented using TensorFlow and Keras, which provide high-level APIs for building and training deep learning models. These frameworks are widely used for developing neural networks such as CNN, RNN, and LSTM due to their efficiency, flexibility, and strong community support.

Natural Language Processing and Embedding Libraries

The project utilizes the Gensim library to implement Word2Vec for generating semantic word embeddings. Word2Vec helps in converting textual data into numerical vectors while preserving contextual relationships between words, which is essential for detecting duplicate questions.

Data Handling and Processing Libraries

Libraries such as NumPy and Pandas are used for data manipulation and preprocessing. These libraries assist in cleaning the dataset, handling missing values, and organizing the data into structured formats suitable for model training.

Machine Learning Utilities

Scikit-learn is used for tasks such as dataset splitting, performance evaluation, and calculation of metrics like accuracy, precision, recall, and F1-score. It provides essential tools for validating the effectiveness of the model.

Backend Framework

For system implementation and possible deployment, frameworks such as Flask or Django can be used to develop the backend. These frameworks enable the creation of APIs that handle user requests and return duplicate detection results efficiently.

Frontend Interface

A simple user interface is developed using Python-based GUI frameworks such as Tkinter or web-based interfaces. This interface allows users to input questions and view duplicate detection results in real time.

Development Environment

The model development and experimentation are carried out using Jupyter Notebook and Google Colab. These environments provide an interactive platform for coding, debugging, and visualizing results, along with optional GPU support for faster training.

Database and Storage

The dataset is stored using file-based storage systems such as CSV files, and optional databases like MySQL or SQLite can be used for managing large-scale data efficiently.

Operating System

The system is developed and executed on a Windows operating system, which provides a stable environment for running the required tools and libraries.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Version Control

Git and GitHub are used for version control and project management. They help in tracking changes, maintaining different versions of the code, and supporting collaborative development.

3.7 DESIGN OF MODULES

Data Input Module

The data input module serves as the entry point of the system, responsible for acquiring all necessary input data. It allows users to upload datasets containing labeled question pairs or submit new queries for duplicate detection. This module ensures that the input data is properly structured, validated, and formatted before it is forwarded to subsequent stages. By handling both batch data and real-time user queries, this module supports flexibility in system usage.

Preprocessing Module

The preprocessing module plays a critical role in transforming raw textual data into a clean and consistent format suitable for analysis. It performs multiple operations including removal of noise such as special characters, HTML tags, and unnecessary symbols, conversion of text to lowercase for uniformity, and tokenization to break sentences into individual words. Additionally, it removes stop words and applies normalization techniques such as stemming or lemmatization. These steps reduce redundancy, improve data quality, and enhance the effectiveness of downstream processing.

Word Embedding Module

The word embedding module converts the preprocessed textual data into dense numerical vector representations using Word2Vec. Unlike traditional methods that treat words independently, this module captures semantic and contextual relationships between words by placing similar words closer in the vector space. This transformation enables the system to understand the meaning of text at a deeper level, making it possible to detect similarities between questions even when they are phrased differently.

Model Processing Module

The model processing module forms the core of the system, where advanced deep learning algorithms are applied to extract meaningful patterns from the embedded data. It utilizes Convolutional Neural Networks to identify local features and important phrases, Recurrent Neural Networks to capture sequential dependencies, and Long Short-Term Memory networks to retain long-term contextual information. By combining these models, the system gains the ability to analyze both structural and semantic aspects of the text, leading to more accurate duplicate detection.

Similarity Computation Module

The similarity computation module evaluates the relationship between two question vectors by applying various comparison techniques. It measures how closely the semantic representations of the questions align, using operations such as vector comparison and similarity scoring. This module is essential in determining the degree of similarity between question pairs, enabling the system to differentiate between identical, similar, and unrelated queries.

Classification Module

The classification module interprets the similarity scores generated by the previous stage and makes the final decision regarding duplication. It uses a trained classification layer to assign a probability score indicating whether the question pair is duplicate or non-duplicate. This probabilistic approach allows for more flexible and accurate predictions, reducing both false positives and false negatives.

Output Module

The output module is responsible for presenting the final results to the user in an understandable format. It displays the classification outcome along with similarity scores and may also provide a ranked list of the most relevant duplicate questions. This enhances user experience by offering meaningful insights and helping users quickly identify existing solutions.

Training and Evaluation Module

The training and evaluation module is responsible for building and validating the deep learning models used in the system. It trains the models using labeled datasets and optimizes them using techniques such as backpropagation and gradient descent. The module

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

also evaluates model performance using metrics such as accuracy, precision, recall, and F1-score. Continuous evaluation ensures that the system maintains high performance and generalizes well to unseen data.

4. IMPLEMENTATION

4.1 MODULE DESCRIPTION

The duplicate question detection system is designed using a modular approach, where each module is responsible for a specific function in the overall workflow. This structured design ensures efficient processing, better organization, and ease of maintenance.

Data Input Module

The data input module is responsible for collecting input data from users or datasets. It accepts question pairs for training as well as individual queries for real-time duplicate detection. This module ensures that the input data is properly structured and formatted for further processing.

Preprocessing Module

The preprocessing module focuses on cleaning and preparing raw textual data. It removes noise such as special characters and unwanted symbols, converts text to lowercase, performs tokenization, and eliminates stop words. These steps improve data consistency and enhance the quality of input for model processing.

Word Embedding Module

The word embedding module converts the preprocessed text into numerical vector representations using Word2Vec. This module captures semantic and contextual relationships between words, enabling the system to understand similarities between questions beyond simple keyword matching.

Model Processing Module

The model processing module applies deep learning algorithms such as Convolutional Neural Networks, Recurrent Neural Networks, and Long Short-Term Memory networks. These models extract meaningful features, analyze sentence structure, and capture contextual dependencies within the text.

Similarity Computation Module

The similarity computation module compares the vector representations of question pairs to determine how closely they are related. It calculates similarity scores based on semantic relationships, which are used to identify potential duplicate questions.

Classification Module

The classification module uses the computed similarity scores to classify question pairs as duplicate or non-duplicate. It generates predictions based on trained model outputs, ensuring accurate and reliable decision-making.

Output Module

The output module presents the final results to the user. It displays whether the questions are duplicates and may also provide a list of the most relevant similar questions for better understanding.

Training and Evaluation Module

The training and evaluation module is responsible for training the deep learning models using labeled datasets and evaluating their performance. It uses metrics such as accuracy, precision, recall, and F1-score to ensure the effectiveness and reliability of the system.

4.2 TRAINING PROCESS FOR AI/ML MODELS

The training process of the proposed duplicate question detection system focuses on enabling deep learning models to accurately identify semantic similarity between question pairs. The pipeline is carefully designed to ensure effective learning, improved generalization, and high prediction accuracy on unseen data. It involves multiple stages, including dataset preparation, preprocessing, embedding generation, model training, optimization, and evaluation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Model Selection

The system employs advanced deep learning models such as Convolutional Neural Networks, Recurrent Neural Networks, and Long Short-Term Memory networks. These models are selected due to their ability to handle textual data efficiently. CNN is effective in extracting local features and identifying important phrases, RNN captures sequential dependencies between words, and LSTM overcomes the limitations of RNN by preserving long-term contextual information. The combination of these models ensures a comprehensive understanding of both structure and meaning in question pairs.

Dataset Preparation

The dataset consists of labeled question pairs obtained from Community-Based Question Answering platforms. Each pair is associated with a binary label indicating whether the questions are duplicates or not. The dataset is preprocessed to remove inconsistencies, normalize text, and ensure quality. It is then divided into training, validation, and testing sets. The training set is used for model learning, the validation set is used to tune hyperparameters and monitor performance, and the testing set is used for final evaluation to measure generalization capability.

Tokenization and Encoding

The textual data is transformed into a machine-readable format through tokenization. This process splits sentences into individual tokens or words, preserving their sequence. After tokenization, the text is converted into numerical representations using embedding techniques. Padding and truncation are applied to ensure that all input sequences have a fixed length, which is required for efficient batch processing in deep learning models. Attention is given to maintaining consistency between training and testing inputs.

Feature Extraction using Word2Vec

Word2Vec is used to generate dense vector representations of words based on their contextual usage within the dataset. Unlike traditional one-hot encoding, Word2Vec captures semantic relationships by placing similar words closer in vector space. This allows the model to understand synonyms, contextual similarities, and paraphrased expressions. These embeddings serve as the input layer for CNN, RNN, and LSTM models, enabling them to process meaningful semantic features.

Model Training

During the training phase, the embedded question pairs are fed into the deep learning models. CNN applies convolutional filters to extract important n-gram features, while pooling layers reduce dimensionality and highlight significant patterns. RNN processes the sequence of words to understand the order and structure of the sentence. LSTM enhances this process by maintaining memory over long sequences, allowing the model to capture contextual dependencies across the entire question. The models learn by identifying patterns that distinguish duplicate questions from non-duplicate ones.

Training Configuration

The training process is controlled using various hyperparameters such as batch size, learning rate, number of epochs, and optimizer selection. A suitable optimizer such as Adam or RMSProp is used to update the model weights efficiently. Learning rate scheduling is applied to adjust the learning rate during training, ensuring stable convergence. Gradient accumulation and batch processing techniques are used to optimize computational efficiency.

Loss Function and Optimization

The models are trained using a binary cross-entropy loss function, which measures the difference between predicted outputs and actual labels. During each training iteration, the model performs forward propagation to generate predictions, calculates the loss, and updates the weights through backpropagation. The objective is to minimize the loss function and improve prediction accuracy over time.

Evaluation During Training

The model performance is continuously monitored using the validation dataset. Metrics such as accuracy, precision, recall, and F1-score are calculated to evaluate the effectiveness of the model. These metrics provide insights into the model's ability to correctly identify duplicate and non-duplicate questions. Validation helps in detecting overfitting and ensures that the model generalizes well to new data.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Handling Training Challenges

During training, several challenges such as overfitting, underfitting, and data imbalance may arise. These are addressed using techniques such as dropout, which randomly deactivates neurons to prevent overfitting, and early stopping, which halts training when validation performance stops improving. Proper preprocessing and balanced datasets further enhance model stability and performance.

Inference and Prediction

After training, the model is used for real-time prediction. When a new question is provided, it is preprocessed and converted into vector form using the same Word2Vec embeddings. The trained model then compares it with existing questions and generates similarity scores. Based on these scores, the system identifies and ranks the most relevant duplicate questions.

Model Saving and Deployment

The final trained model, along with embedding configurations and preprocessing parameters, is saved for deployment. The model can be integrated into a real-time application or API, allowing users to input queries and receive duplicate detection results instantly. This ensures practical usability and scalability of the system.

4.3 SCREENSHOTS

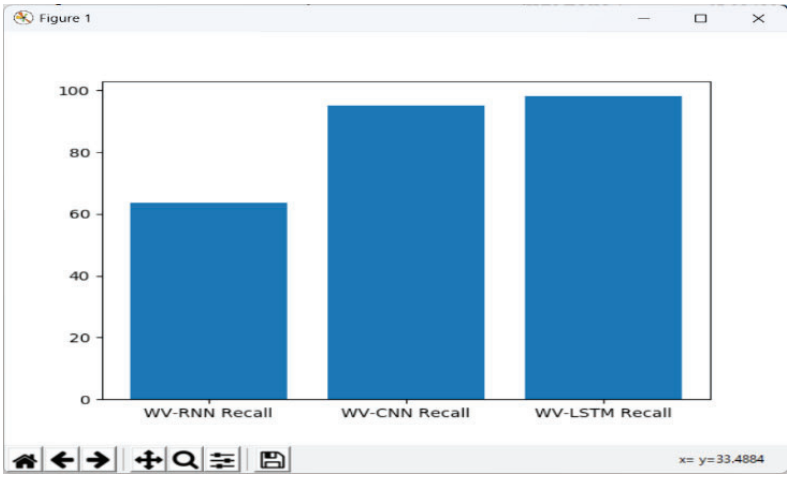

FIGURE 4.3.1: Recall Graph

The graph presents a visual comparison of recall values for the three deep learning models used in the system, namely RNN, CNN, and LSTM. Recall is an important evaluation metric that measures the ability of the model to correctly identify actual duplicate questions.

From the graph, it can be observed that the LSTM model achieves the highest recall, indicating its strong capability in capturing long-term dependencies and contextual relationships in text data. The CNN model also demonstrates high recall performance, showing its effectiveness in extracting important features and patterns from the input data. In contrast, the RNN model has a lower recall value, which highlights its limitations in handling long sequences and retaining contextual information.

Overall, the graph clearly shows that LSTM outperforms the other models in terms of recall, making it the most suitable model for duplicate question detection in this system.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

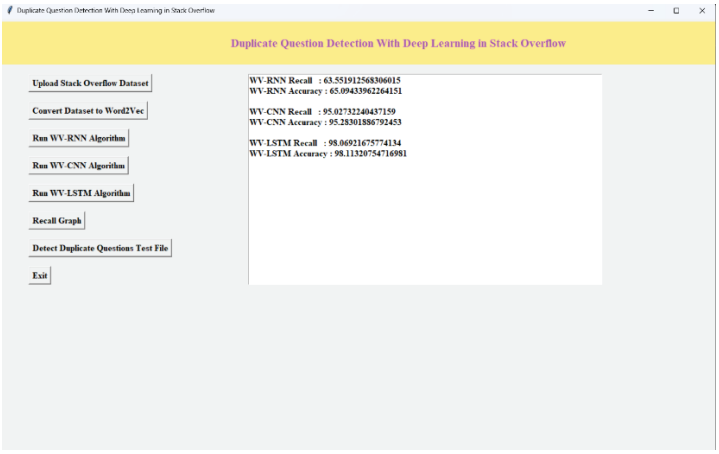

Figure 4.3.2 : Duplicate Question Detection System Interface

The screenshot illustrates the graphical user interface of the duplicate question detection system developed using a Python-based GUI framework. The interface provides multiple functional options such as uploading the Stack Overflow dataset, converting the dataset into Word2Vec embeddings, executing deep learning models including RNN, CNN, and LSTM, generating recall graphs, and detecting duplicate questions from a test file.

On the right side, the system displays the performance metrics of each model, including recall and accuracy values. The results indicate that the LSTM model achieves the highest performance, followed by CNN, while RNN shows comparatively lower accuracy and recall. This interface enables users to interact with the system easily, execute different modules step-by-step, and observe the model performance in real time, making it user-friendly and efficient for analysis.

4.4 SMPLE INPUT/OUTPUT

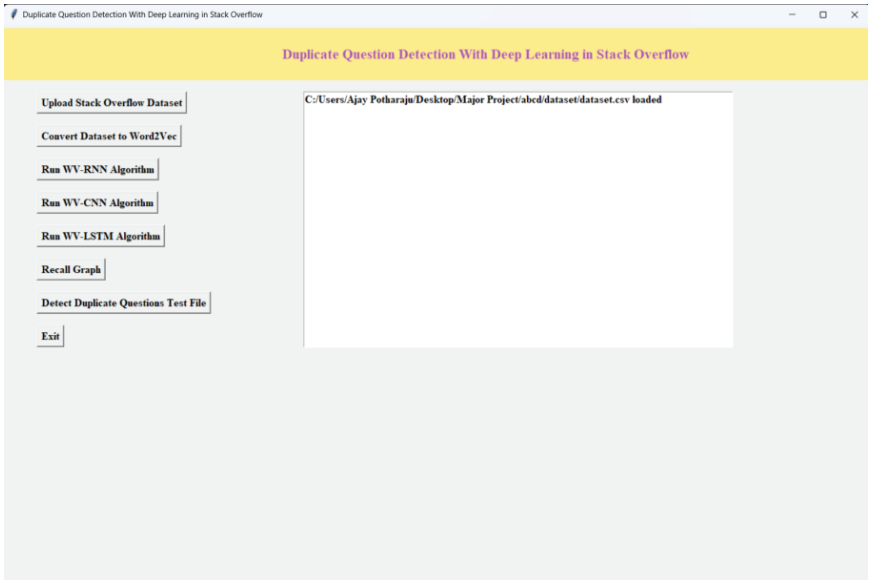

Figure 4.4.1 : Sample Input

The input section is located on the left side of the interface and consists of multiple functional buttons that allow the user to interact with the system. The primary input begins with the **Upload Stack Overflow Dataset** option, where a dataset containing question pairs is provided to the system. This dataset serves as the foundational input for training and testing the models.

The next input operation is **Convert Dataset to Word2Vec**, which transforms the textual dataset into numerical vector representations. This step prepares the data for processing by deep learning models by capturing semantic relationships between words.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Subsequently, the user can initiate different model executions using the options **Run WV-RNN Algorithm**, **Run WV-CNN Algorithm**, and **Run WV-LSTM Algorithm**. These inputs trigger the respective deep learning models to process the dataset and learn patterns for duplicate detection.

The **Recall Graph** option allows the user to visualize model performance, while the **Detect Duplicate Questions Test File** button enables the system to process new or unseen questions. This step acts as the final input stage where real-time queries are analyzed to detect duplicates.

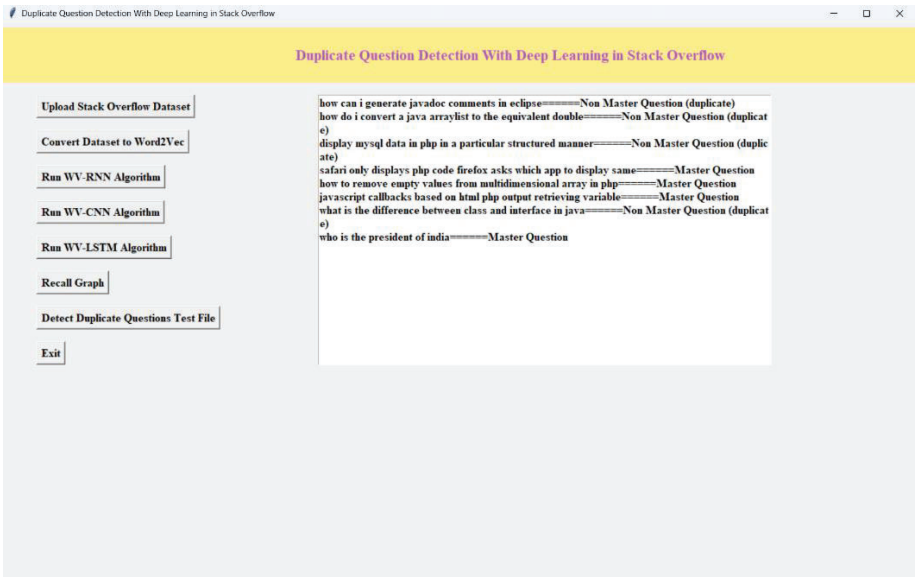

Figure 4.4.2 : Sample Output

The output section is displayed on the right side of the interface and presents the results of the duplicate question detection process. Each line in the output corresponds to a question from the test dataset, followed by its classification result.

The system categorizes questions into two main types: **Master Question** and **Non-Master Question (duplicate)**. A master question represents the original or unique version of a query, while a non-master question indicates a duplicate or semantically similar question already present in the dataset.

For example, certain technical questions related to programming topics are identified as duplicates, demonstrating the model's ability to recognize semantic similarity even when the wording differs. On the other hand, distinct or unrelated questions are classified as master questions, indicating that they are unique and not duplicates.

The output clearly reflects the effectiveness of the deep learning models in distinguishing between duplicate and non-duplicate questions. By providing structured and readable results, the system enables users to easily interpret the classification outcomes.

5. RESULT AND DISCUSSION

5.1 PERFORMANCE ANALYSIS

The performance analysis of the proposed duplicate question detection system is conducted to evaluate its effectiveness in identifying semantically similar question pairs in Community-Based Question Answering platforms. The analysis focuses on measuring the accuracy, recall, reliability, and practical usefulness of the system by comparing the performance of three deep learning models, namely Word2Vec-based Recurrent Neural Network, Word2Vec-based Convolutional Neural Network, and Word2Vec-based Long Short-Term Memory Network. The uploaded paper and the result screenshots together show that the models were trained and tested under the same setup, allowing a fair comparison of their performance.

Evaluation Dataset

The model is evaluated using a labeled dataset of question pairs collected from Community-Based Question Answering platforms such as Stack Overflow. Each record contains two questions and a binary label indicating whether the pair is duplicate or non-duplicate. According to the uploaded paper, the dataset contains diverse question pairs with variations in wording, structure, and context, which is important for testing semantic understanding rather than simple word matching. The dataset is divided using an

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

80:20 split, where 80% of the data is used for training and 20% is reserved for testing and performance evaluation. This separation helps ensure an unbiased assessment of model performance on unseen data.

PERFORMANCE METRICS

To assess the proposed system, the following metrics are considered in the project documentation and result analysis.

Accuracy
Accuracy measures the overall percentage of question pairs that are classified correctly as duplicate or non-duplicate. It gives a general indication of how well the model performs across the entire test set. In this project, accuracy is one of the main metrics used to compare the three deep learning models.

Recall
Recall measures the ability of the model to correctly identify actual duplicate questions. This metric is especially important in duplicate detection tasks because failing to detect a duplicate means redundant content remains in the platform. A higher recall indicates that the model is more effective in capturing semantically similar and paraphrased questions.

Observed Results

Based on the uploaded screenshots and the results section of the PDF, the system achieved the following performance.

- WV-RNN

  Recall ≈ 63.55%

  Accuracy ≈ 65.09%

- WV-CNN

  Recall ≈ 95.03%

  Accuracy ≈ 95.28%

- WV-LSTM

  Recall ≈ 98.06%

  Accuracy ≈ 98.11%

These values are also visible in the GUI screenshot, where the model execution panel displays recall and accuracy for WV-RNN, WV-CNN, and WV-LSTM. The recall graph screenshot further confirms the same ranking order, with WV-LSTM as the best-performing model, followed by WV-CNN and WV-RNN. The testing-output screenshot additionally shows that the system is capable of labeling questions as “Master Question” or “Non Master Question (duplicate),” demonstrating practical duplicate detection behavior beyond metric values alone.

Analysis of Results

The results clearly show that the WV-LSTM model achieves the best performance among all three evaluated models. With a recall of 98.06% and an accuracy of 98.11%, LSTM demonstrates a strong ability to understand semantic similarity between question pairs, even when the wording differs. The paper attributes this superior performance to the memory cells and gating mechanisms of LSTM, which allow it to retain important contextual information over longer text sequences. This makes LSTM especially suitable for duplicate question detection, where meaning is often spread across multiple words or phrases rather than being visible through direct word overlap.

The WV-CNN model also performs very well, achieving 95.03% recall and 95.28% accuracy. This indicates that CNN is highly effective in extracting local textual patterns, such as repeated phrases, important keywords, and short semantic structures. The result suggests that convolution-based feature extraction is useful for duplicate detection, particularly when similar wording patterns appear in the questions. However, compared with LSTM, CNN is relatively less effective in capturing broader sequential and long-range contextual dependencies. This explains why its performance, while strong, still remains below that of WV-LSTM.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

The WV-RNN model shows the lowest performance, with 63.55% recall and 65.09% accuracy. This comparatively weak result suggests that the simple RNN architecture is not sufficient for modeling complex semantic relationships in question pairs. As noted in the uploaded paper, RNN suffers from limitations in handling long-term dependencies and is affected by the vanishing gradient problem. Because duplicate questions are often paraphrased and may require context from the full sequence, the inability of RNN to preserve long-range information reduces its effectiveness substantially.

The recall graph shown in the screenshot visually supports this analysis. The bar chart clearly shows a major gap between WV-RNN and the other two models, while WV-LSTM slightly exceeds WV-CNN. This graphical comparison makes it evident that the addition of stronger contextual modeling directly improves duplicate detection performance. In other words, as the model's ability to retain semantic context improves, its recall also improves. That relationship is clearly reflected in the progression from RNN to CNN to LSTM.

The testing-results screenshot further strengthens the analysis by showing the output of the deployed system on sample questions. In that interface, some questions are marked as "Master Question," while others are labeled "Non Master Question (duplicate)." This indicates that the trained model is not only strong in offline metric evaluation but is also capable of producing understandable classification results in the user-facing application. This practical usability is important because it demonstrates that the model can support real-time duplicate detection workflows in an actual software interface.

Comparison with Baseline Approaches

The uploaded documents explain that traditional duplicate detection methods rely on lexical similarity, term overlap, cosine similarity, WordNet similarity, and other handcrafted features. These approaches are limited in capturing paraphrased and contextually equivalent questions. In contrast, the proposed system combines Word2Vec embeddings with deep learning models, which allows it to understand semantic relationships beyond exact word matching. Based on the large improvement seen in CNN and LSTM results, it can be inferred that the proposed neural approach substantially outperforms simpler rule-based or shallow lexical techniques for this task.

System Efficiency and Practical Utility

The screenshots suggest that the system has been implemented with a graphical user interface that allows the user to upload the dataset, convert it into Word2Vec embeddings, run the three deep learning models, view recall graphs, and test duplicate detection on sample questions. This modular execution flow indicates that the system is practical and usable for experimentation as well as demonstration. The documents also state that the interface is meant to provide real-time duplicate detection results and improve user experience by making duplicate identification faster and easier.

Limitations Identified

Although the results are strong, the uploaded paper also points out several limitations. The dataset may not fully represent all real-world scenarios, especially where questions differ greatly in domain, structure, or language. The models also require significant computational resources during training. In addition, misclassification may still occur when questions share similar keywords but differ in intent, or when semantically identical questions have very low lexical overlap. These limitations are important because they show that even a high-performing LSTM model can still face challenges in more complex and diverse environments.

5.2 EVALUATION METRICS SUCH AS ACCURACY, PRECISION, RECALL, F1-SCORE,CONFUSION MATRIX

1. ACCURACY

Accuracy measures the proportion of correctly predicted question pairs out of the total number of samples. It provides an overall evaluation of how well the model performs in classifying duplicate and non-duplicate questions.

Accuracy is calculated as:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

In this system, accuracy is used to compare the performance of different deep learning models such as WV-RNN, WV-CNN, and WV-LSTM. Higher accuracy indicates better overall classification performance.

2. PRECISION

Precision measures the proportion of correctly predicted duplicate questions out of all predicted duplicate cases. It reflects how many of the predicted duplicates are actually correct.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Precision = TP / (TP + FP)

Where,

TP represents correctly identified duplicate questions

FP represents incorrectly identified duplicate questions

High precision indicates that the model produces fewer false duplicate predictions.

3. RECALL

Recall measures the proportion of correctly identified duplicate questions out of all actual duplicate questions. It reflects how effectively the model captures all relevant duplicate pairs.

Recall = TP / (TP + FN)

Where,

FN represents duplicate questions that were not identified by the model

High recall indicates that the model successfully detects most of the duplicate questions. In this project, recall is a key metric because missing duplicates leads to redundancy in the system.

4. F1-SCORE

The F1-score is the harmonic mean of precision and recall, providing a balanced evaluation of the model's performance.

F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

This metric is particularly useful when there is an imbalance between duplicate and non-duplicate question pairs, ensuring that both false positives and false negatives are considered.

5. EXACT MATCH ACCURACY

This metric measures the percentage of cases where the model correctly classifies the question pair exactly as duplicate or non-duplicate. It provides a strict evaluation of classification performance and reflects the model's overall correctness.

6. CLASS-LEVEL ACCURACY

This metric evaluates how accurately the model classifies each class separately, namely duplicate and non-duplicate questions. It helps in understanding whether the model performs equally well for both categories.

7. PREDICTION CONSISTENCY

This metric evaluates the stability of the model's predictions across different inputs. It ensures that similar question pairs consistently produce similar outputs, which is important for maintaining reliability in real-world applications.

5.3 TEST CASES

The following test cases are designed to validate the functionality, accuracy, and reliability of the duplicate question detection system. These test cases ensure that each module of the system performs correctly, from dataset upload to final duplicate classification.

TEST CASE 1

Test Case Name: Dataset Upload Verification

Objective: To verify whether the system successfully uploads the Stack Overflow dataset.

Input: Valid dataset file containing question pairs.

Expected Output: The dataset should be uploaded successfully and made available for further processing.

Actual Output: The dataset is uploaded successfully.

Result: Pass

TEST CASE 2

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Test Case Name: Word2Vec Conversion Verification

Objective: To verify whether the uploaded dataset is correctly converted into Word2Vec embeddings.

Input: Pre-processed Stack Overflow dataset.

Expected Output: The system should generate vector representations for the textual data without errors.

Actual Output: The dataset is successfully converted into Word2Vec embeddings.

Result: Pass

TEST CASE 3

Test Case Name: WV-RNN Model Execution

Objective: To verify whether the WV-RNN algorithm runs successfully and generates performance metrics.

Input: Word2Vec embedded dataset.

Expected Output: The WV-RNN model should execute successfully and display recall and accuracy values.

Actual Output: The system displays WV-RNN Recall = 63.55% and Accuracy = 65.09%.

Result: Pass

TEST CASE 4

Test Case Name: WV-CNN Model Execution

Objective: To verify whether the WV-CNN algorithm runs successfully and generates performance metrics.

Input: Word2Vec embedded dataset.

Expected Output: The WV-CNN model should execute successfully and display recall and accuracy values.

Actual Output: The system displays WV-CNN Recall = 95.03% and Accuracy = 95.28%.

Result: Pass

TEST CASE 5

Test Case Name: WV-LSTM Model Execution

Objective: To verify whether the WV-LSTM algorithm runs successfully and generates performance metrics.

Input: Word2Vec embedded dataset.

Expected Output: The WV-LSTM model should execute successfully and display recall and accuracy values.

Actual Output: The system displays WV-LSTM Recall = 98.06% and Accuracy = 98.11%.

Result: Pass

TEST CASE 6

Test Case Name: Recall Graph Generation

Objective: To verify whether the system generates the recall comparison graph correctly.

Input: Recall values of WV-RNN, WV-CNN, and WV-LSTM models.

Expected Output: A bar graph should be displayed comparing recall values of all three models.

Actual Output: The graph is generated successfully and shows LSTM with the highest recall.

Result: Pass

TEST CASE 7

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Test Case Name: Duplicate Question Detection from Test File

Objective: To verify whether the system correctly identifies duplicate and non-duplicate questions from the test file.

Input: Test file containing new question queries.

Expected Output: The system should classify each question as Master Question or Non-Master Question (duplicate).

Actual Output: The system displays the classification results correctly for the input questions.

Result: Pass

TEST CASE 8

Test Case Name: Unique Question Identification

Objective: To verify whether the system correctly identifies a unique question as non-duplicate.

Input: A question with no semantically similar match in the dataset.

Expected Output: The system should classify the question as a Master Question.

Actual Output: The question is classified as Master Question.

Result: Pass

TEST CASE 9

Test Case Name: Duplicate Question Identification

Objective: To verify whether the system correctly identifies a duplicate question.

Input: A question semantically similar to an existing question in the dataset.

Expected Output: The system should classify the question as Non-Master Question (duplicate).

Actual Output: The question is classified as Non-Master Question (duplicate).

Result: Pass

TEST CASE 10

Test Case Name: System Exit Verification

Objective: To verify whether the system closes properly when the Exit button is selected.

Input: Exit command from user interface.

Expected Output:The application should terminate successfully without errors.

Actual Output: The application closes successfully.

Result: Pass

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 FINAL CONCLUSION

This project presents a deep learning-based approach for duplicate question detection in Community-Based Question Answering platforms. The primary objective of the system is to identify semantically similar questions and reduce redundancy in large-scale datasets such as Stack Overflow. By integrating Word2Vec embeddings with deep learning models including Convolutional Neural Networks, Recurrent Neural Networks, and Long Short-Term Memory networks, the system effectively captures both semantic meaning and contextual relationships within textual data.

The experimental results demonstrate that the proposed system significantly improves duplicate detection performance compared to traditional methods based on lexical similarity. Among the evaluated models, the WV-LSTM model achieved the highest performance, with a recall of approximately 98.06% and an accuracy of 98.11%, indicating its strong capability in capturing long-term dependencies and understanding complex sentence structures. The WV-CNN model also performed well, achieving high accuracy and recall by effectively extracting important textual features. In contrast, the WV-RNN model showed comparatively

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

lower performance due to limitations such as the vanishing gradient problem and its inability to retain long-range contextual information.

The system not only performs well in terms of evaluation metrics but also demonstrates practical usability through its graphical user interface. The interface allows users to upload datasets, execute models, visualize performance through graphs, and detect duplicate questions in real time. This makes the system suitable for both research and real-world applications.

Overall, the proposed framework provides an efficient, scalable, and reliable solution for duplicate question detection. It enhances information retrieval, reduces redundancy, and improves knowledge organization in CQA platforms. The integration of semantic embeddings and deep learning models enables the system to detect duplicates even when questions are expressed differently, addressing a key challenge in natural language processing.

In conclusion, the project successfully demonstrates the effectiveness of deep learning techniques for semantic similarity detection and provides a strong foundation for further advancements in intelligent question-answering systems.

6.2 LIMITATIONS

Despite achieving high accuracy and recall, the proposed duplicate question detection system has certain limitations that need to be addressed for real-world deployment and scalability.

1. Dependence on Dataset Quality and Domain Coverage

The performance of the system is highly dependent on the quality, size, and diversity of the training dataset. The dataset used in this project primarily consists of question pairs from specific domains such as programming and technical discussions. As a result, the model may not generalize well to other domains such as healthcare, legal queries, or general knowledge questions. Limited exposure to diverse linguistic patterns, writing styles, and domain-specific vocabulary can lead to reduced performance when handling unseen or heterogeneous data.

2. Limited Contextual Understanding with Word2Vec

The system relies on Word2Vec embeddings for feature representation, which capture semantic similarity based on surrounding words. However, Word2Vec generates static embeddings, meaning a word has the same vector representation regardless of its context. This limits the system's ability to handle polysemy, where a word may have multiple meanings depending on context. Consequently, the model may misinterpret questions where contextual meaning plays a critical role.

3. Inability to Fully Capture Deep Semantic Relationships

Although deep learning models such as CNN, RNN, and LSTM improve semantic understanding, they still have limitations in handling complex linguistic phenomena such as sarcasm, idiomatic expressions, implicit intent, and long-distance dependencies across multiple sentences. Questions that are semantically identical but structurally very different may not always be detected as duplicates.

4. Model Limitations in Handling Ambiguity and Edge Cases

The system may produce incorrect predictions in cases where questions share similar keywords but differ in intent. For example, two questions may use identical technical terms but refer to entirely different problems. Similarly, questions with very low lexical similarity but identical intent may not be detected as duplicates. These edge cases highlight the challenge of distinguishing between surface-level similarity and true semantic equivalence.

5. Computational Complexity and Resource Requirements

The training of deep learning models such as CNN, RNN, and LSTM requires significant computational resources, including high memory usage and processing power. Training multiple models further increases computational cost and time. This makes it challenging to deploy the system in resource-constrained environments or to retrain the model frequently with updated data.

6. Training Time and Scalability Issues

The system requires considerable time for training, especially when working with large datasets. As the dataset size increases, training time grows significantly, which may hinder scalability. Additionally, real-time retraining or continuous learning is not feasible with the current architecture.

7. Lack of Real-Time Adaptability

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

The current system operates using a pre-trained model and does not support dynamic learning. It cannot automatically adapt to new trends, evolving language patterns, or newly emerging topics without retraining. This limits its ability to remain relevant in rapidly changing environments.

8. Binary Classification Limitation

The system is designed as a binary classifier that labels question pairs as either duplicate or non-duplicate. It does not provide a graded similarity score or confidence level in a detailed manner. This limits its ability to support advanced applications such as ranking multiple similar questions or recommending top-K relevant results.

9. Limited Multilingual and Cross-Lingual Support

The system is primarily designed for English text and does not support multilingual or cross-lingual duplicate detection. In real-world applications, users may post questions in multiple languages, which the current system cannot effectively handle.

10. User Interface and Deployment Constraints

Although the system provides a graphical user interface for demonstration purposes, it is not fully optimized for large-scale deployment. The current interface may not efficiently handle high user traffic, concurrent requests, or integration with large-scale CQA platforms.

6.3 POSSIBLE FUTURE ENHANCEMENTS

To overcome the identified limitations and further enhance the system's performance, scalability, and applicability, several improvements can be implemented.

1. Integration of Transformer-Based Models

Future enhancements can include the use of advanced transformer-based models such as BERT, RoBERTa, or GPT. These models generate contextual embeddings, where the meaning of a word depends on its surrounding context. This will significantly improve the system's ability to capture deep semantic relationships, handle ambiguity, and detect paraphrased questions more accurately.

2. Development of Hybrid and Ensemble Models

Combining multiple architectures such as CNN, LSTM, and transformer models can create a hybrid system that leverages the strengths of each approach. Ensemble learning techniques can also be applied to combine predictions from multiple models, improving overall accuracy and robustness.

3. Expansion of Dataset and Domain Diversity

The system can be improved by training on larger and more diverse datasets that include multiple domains, writing styles, and linguistic variations. Incorporating real-world datasets from different sources will enhance the model's generalization capability and make it more robust in practical applications.

4. Implementation of Semantic Similarity Ranking

Instead of performing only binary classification, the system can be enhanced to generate similarity scores and rank the top K most relevant duplicate questions. This will improve usability by providing users with multiple relevant suggestions rather than a single classification result.

5. Real-Time Learning and Continuous Model Updating

Incorporating online learning or incremental training techniques will allow the model to update dynamically as new data becomes available. This will help the system adapt to evolving user behavior, new terminology, and changing trends.

6.Optimization for Computational Efficiency

Techniques such as model compression, pruning, and quantization can be applied to reduce model size and computational requirements. This will improve inference speed and make the system suitable for real-time deployment on low-resource devices.

7. Multilingual and Cross-Lingual Support

Future work can extend the system to support multiple languages using multilingual embeddings or translation-based approaches. This will make the system applicable to global platforms and diverse user bases.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

8. Integration with Real-World Platforms

The system can be integrated into live CQA platforms such as Stack Overflow or Quora. It can be used to suggest duplicate questions during query submission, reducing redundancy and improving user experience.

9. Incorporation of Attention Mechanisms

Adding attention layers to the model can help identify important words and phrases that contribute most to duplicate detection. This will improve interpretability and provide insights into model decision-making.

10. Improved User Interface and Scalability

The graphical user interface can be enhanced into a web-based application with scalable backend architecture. Technologies such as cloud deployment and REST APIs can be used to handle large-scale user interactions efficiently.

7. REFERENCES

BIBLIOGRAPHY

[1] Correa, D., & Sureka, A. “Chaff from the wheat: Characterization and modeling of deleted questions on community-based question answering websites”, Proceedings of the 23rd International Conference on World Wide Web, pp. 631–642 (2014).

[2] Zhang, Y., Lo, D., Xia, X., & Sun, J.-L. “Multi-factor duplicate question detection in community question answering platforms”, Journal of Computer Science and Technology, 30(5), pp. 981–997 (2015).

[3] Silva, R. F., Paixão, K., & de Almeida Maia, M. “Duplicate question detection: A reproducibility study”, Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 572–581 (2018).

[4] Kamath, C. N., Bukhari, S. S., & Dengel, A. “Comparative study between traditional machine learning and deep learning approaches for text classification”, Proceedings of the ACM Symposium on Document Engineering, pp. 1–11 (2018).

[5] Mihalcea, R., Corley, C., & Strapparava, C. “Corpus-based and knowledge-based measures of text semantic similarity”, Proceedings of AAAI, pp. 775–780 (2006).

[6] Bangalore, S., & Sangal, R. “Question classification using support vector machines”, International Conference on Natural Language Processing (ICON), pp. 1–8 (2011).

[7] Bogdanova, D., Florou, E., & Pallotti, S. “Detecting semantically equivalent questions in community question answering forums”, Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), pp. 1–7 (2015).

APPENDIX / SOURCE CODE

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from tkinter import filedialog

from sklearn.feature_extraction.text import CountVectorizer

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad_sequences

from nltk.corpus import stopwords

import re
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

```
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score, accuracy_score
from keras.layers import Activation, Dropout
from keras.layers.convolutional import Conv1D
from keras.layers import GlobalMaxPooling1D
main = tkinter.Tk()
main.title('Duplicate Question Detection With Deep Learning in Stack Overflow')
main.geometry("1300x1200")
global filename, model
global rnn_recall, cnn_recall, lstm_recall
global X, Y, tokenizer
global X_train, X_test, y_train, y_test
stop_words = set(stopwords.words('english'))
def rem_html_tags(question):
    regex = re.compile('<.*?>')
    return re.sub(regex, '', question)
def removeSpecialSymbols(question):
    question = re.sub('\W+', ' ', question)
    return question.strip()
def upload():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="dataset")
    text.insert(END, filename + " loaded\n")
def word2Vec():
    global tokenizer, X, Y
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    train = pd.read_csv(filename, encoding='iso-8859-1', nrows=50000)
    X, Y = [], []
    nondup = 0
    for i in range(len(train)):
        title = rem_html_tags(train._get_value(i, 'Title'))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

```
        title = removeSpecialSymbols(title).lower()
        body = train._get_value(i, 'Body')
        body = removeSpecialSymbols(body.lower())
        label = 1 if 'possible duplicate' in body else 0
        words = title.split()
        filtered = [w for w in words if len(w) > 2 and w not in stop_words]
        text_data = " ".join(filtered)
        if label == 1:
            X.append(text_data)
            Y.append(label)
        elif nondup <= 300:
            X.append(text_data)
            Y.append(label)
            nondup += 1
    X = np.array(X)
    Y = to_categorical(np.array(Y))
    tokenizer = Tokenizer(num_words=500)
    tokenizer.fit_on_texts(X)

    X = tokenizer.texts_to_sequences(X)
    X = pad_sequences(X)
    X_train, X_test, y_train, y_test = train_test_split(
        X, Y, test_size=0.2, random_state=0
    )
    text.insert(END, f"Total Questions: {len(X)}\n")
    text.insert(END, f"Training: {X_train.shape[0]}\n")
    text.insert(END, f"Testing: {X_test.shape[0]}\n")
def WVRNN():
    global rnn_recall
    text.delete('1.0', END)
    model = Sequential()
    model.add(Dense(512, input_shape=(X.shape[1],), activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(2, activation='softmax'))
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

```
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X, Y, epochs=100, batch_size=256)
    y_pred = np.argmax(model.predict(X_test), axis=1)
    y_true = np.argmax(y_test, axis=1)
    recall = recall_score(y_true, y_pred, average='macro') * 100
    accuracy = accuracy_score(y_true, y_pred) * 100
    text.insert(END, f"WV-RNN Recall: {recall}\n")
    text.insert(END, f"WV-RNN Accuracy: {accuracy}\n")
    rnn_recall = recall
def WVCNN():
    global cnn_recall
    model = Sequential()
    model.add(Embedding(500, 100, input_length=X.shape[1]))
    model.add(Conv1D(128, 5, activation='relu'))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(2, activation='sigmoid'))
    model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accuracy'])
    model.fit(X, Y, epochs=100, batch_size=256)
    y_pred = np.argmax(model.predict(X_test), axis=1)
    y_true = np.argmax(y_test, axis=1)
    recall = recall_score(y_true, y_pred, average='macro') * 100
    accuracy = accuracy_score(y_true, y_pred) * 100
    text.insert(END, f"WV-CNN Recall: {recall}\n")
    text.insert(END, f"WV-CNN Accuracy: {accuracy}\n")
    cnn_recall = recall
def WVLSTM():
    global lstm_recall
    model = Sequential()
    model.add(Embedding(500, 70, input_length=X.shape[1]))
    model.add(SpatialDropout1D(0.4))
    model.add(LSTM(70, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(2, activation='softmax')
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X, Y, epochs=100, batch_size=256)
    y_pred = np.argmax(model.predict(X_test), axis=1)
    y_true = np.argmax(y_test, axis=1)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

```
    recall = recall_score(y_true, y_pred, average='macro') * 100
    accuracy = accuracy_score(y_true, y_pred) * 100
    text.insert(END, f"WV-LSTM Recall: {recall}\n")
    text.insert(END, f"WV-LSTM Accuracy: {accuracy}\n")
    lstm_recall = recall
def detectDuplicates():
    text.delete('1.0', END)
    fname = filedialog.askopenfilename(initialdir="dataset")
    testfile = pd.read_csv(fname, encoding='iso-8859-1')
    for i in range(len(testfile)):
        body = testfile._get_value(i, 'question')
        seq = tokenizer.texts_to_sequences([body])
        seq = pad_sequences(seq, maxlen=10)
        result = np.argmax(model.predict(seq))
        label = "Non Master Question (duplicate)" if result == 1 else "Master Question"
        text.insert(END, f"{body} ====== {label}\n")
def recallGraph():
    plt.bar(['RNN', 'CNN', 'LSTM'], [rnn_recall, cnn_recall, lstm_recall])
    plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Duplicate Question Detection')
title.config(font=font)
title.place(x=0, y=5)
text = Text(main, height=20, width=80)
text.place(x=450, y=100)
Button(main, text="Upload Dataset", command=upload).place(x=50, y=100)
Button(main, text="Word2Vec", command=word2Vec).place(x=50, y=150)
Button(main, text="RNN", command=WVRNN).place(x=50, y=200)
Button(main, text="CNN", command=WVCNN).place(x=50, y=250)
Button(main, text="LSTM", command=WVLSTM).place(x=50, y=300)
Button(main, text="Recall Graph", command=recallGraph).place(x=50, y=350)
Button(main, text="Detect", command=detectDuplicates).place(x=50, y=400)
main.mainloop()
```