

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

# CAPSTONE PROJECT REPORT

**PROJECT TITLE**

**Converstional Bots: ChatBot Using CNN**

# REPORT SUBMITTED BY

G PAVAN KUMAR (192210610)

D RAHEEM (192210537)

S MOHAMMAD TAJ (192211964)

**REPORT SUBMITTED TO**

Dr. LATHA

# COURSE CODE / COURSE NAME

CSA1370 / THEORY OF COMPUTATION WITH PROBLEM SOLVING

**SLOT D**

# ABSTRACT

Chatbots have become crucial in providing automated customer service, handling queries, and enhancing user engagement. Traditionally, chatbots used rule-based or machine learning models to understand and generate responses. However, with advancements in deep learning, Convolutional Neural Networks (CNNs) have emerged as a powerful tool to improve chatbot accuracy by processing textual data more efficiently. This project explores the use of CNNs in building a chatbot that can understand and respond to user inputs. We focus on text preprocessing, feature extraction, and the development of a CNN model for language understanding, aiming for high performance in response generation.

# INTRODUCTION

Conversational agents or chatbots are systems designed to simulate human conversation through text or speech. They are widely used in customer support, personal assistants, and automated systems for various industries. Chatbots typically require the ability to understand natural language input and generate coherent responses. Traditional approaches to building chatbots include rule-based models, sequence-to-sequence models, and machine learning techniques like Naive Bayes or Support Vector Machines (SVM).

With the evolution of deep learning, CNNs have been applied successfully to NLP tasks. CNNs, which excel in pattern recognition, can be trained to understand word embeddings and extract features from text, allowing chatbots to better interpret user inputs. This project aims to develop a chatbot utilizing CNN to enhance its language understanding capabilities and provide more accurate responses to users.

# RESEARCH PLAN

### Literature Review

We will review existing literature on chatbot development, particularly those that have utilized CNNs. This includes works focused on natural language processing (NLP), deep learning models, and their applications in conversational agents.

### Data Collection and Preprocessing

We will use publicly available datasets for chatbot conversations (e.g., Cornell Movie Dialogues dataset) and real-world customer interaction data. Preprocessing will include cleaning, tokenizing, and converting text into numerical representations such as word embeddings (Word2Vec, GloVe).

### Feature Extraction

We will experiment with various feature extraction techniques, such as n-grams and word embeddings, to identify the best features that can be fed into the CNN model. We will explore different architectures to optimize the chatbot's understanding capabilities.

### Model Selection

For the chatbot, we will evaluate CNN-based models compared to other deep learning architectures like Recurrent Neural Networks (RNN) and Transformer models. We aim to build a CNN architecture that efficiently handles conversation context and generates accurate responses.

### Hyperparameter Tuning and Optimization

We will optimize our model using techniques such as grid search and cross-validation. Hyperparameters like the number of convolutional layers, kernel size, and dropout rate will be adjusted to improve performance.

### Evaluation Metrics

We will evaluate the chatbot's performance using accuracy, precision, recall, F1-score, and BLEU (Bilingual Evaluation Understudy Score) for response quality. These metrics will be used to assess how well the chatbot understands user inputs and generates relevant responses.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.NO | DESCRIPTION | 04.09.2024  DAY-01 | 05.09.2024  DAY-02 | 06.09.2024  DAY-03 | 09.09.2024  DAY-04 | 10.09.2024  DAY-05 |
| 1. | Project Initiation and Planning |  |  |  |  |  |
| 2. | Requirement Analysis and Design |  |  |  |  |  |
| 3. | Development and Implementation |  |  |  |  |  |
| 4. | Testing and Refinement |  |  |  |  |  |
| 5. | Documentation, Deployment, and Feedback |  |  |  |  |  |

**Fig. 1 Timeline chart**

**Day 1: Project Initiation and Planning (1 day)**

* Objectives: Develop a spam classification system that accurately detects spam messages.
* Scope: Focus on text-based spam across various platforms (e.g., email, SMS).
* Stakeholders: Identify and engage key stakeholders like team members, experts, and users.
* Resources: Allocate necessary resources, including people, tools, and budget.
* Timeline: Create a timeline with milestones for tasks like data collection, model training, and deployment.
* Risks: Identify potential risks (like data issues) and plan how to handle them.
* Communication: Set up regular communication with the team and stakeholders.
* Success Criteria: Define what success looks like (e.g., model accuracy, timely delivery).
* Kick-off: Hold a project kick-off meeting to start the work.
* Approval: Get approval from all key stakeholders to begin.

**Day 2: Requirement Analysis and Design (1 day)**

**Requirements:**

* System should detect spam accurately.
* Must handle multiple types of messages (email, SMS).
* Needs to be fast, scalable, and secure.
* Use existing machine learning tools (like Python libraries).

**Design:**

* Create a data pipeline for collecting and cleaning data.
* Choose and train the best model for spam detection.
* Set up evaluation criteria (accuracy, precision).
* Plan for deployment in real-world applications..

**Day 3: Development and Implementation (2 days)**

**Develop the System:**

* Build the data pipeline for collecting, cleaning, and processing data.
* Implement the chosen machine learning model(s) for spam detection.
* Train the model using the prepared datasets.

**Test and Validate:**

* Evaluate the model’s performance using test data.
* Adjust and fine-tune the model to improve accuracy and efficiency.

**Deploy the System:**

* Deploy the model in the target environment (e.g., email server, messaging app).
* Monitor the system for performance and update as needed.

**Day 4: Testing and Refinement (1 day)**

**Testing:**

* Test the system with different datasets to check accuracy and reliability.
* Use metrics like accuracy, precision, and recall to evaluate performance.

**Refinement:**

* Identify any errors or weaknesses in the model.
* Adjust the model, retrain if needed, and improve its performance.

**Day 5: Documentation, Deployment, and Feedback (1 day)**

**Documentation:**

* Create clear documentation on system design, development, and usage.

**Deployment:**

* Deploy the system in the intended environment (e.g., email or messaging app).

**Feedback:**

* Collect feedback from users and stakeholders.
* Make improvements based on the feedback received.

# MATERIALS AND METHODS

**Materials:**

Datasets: Dialogue datasets (e.g., Cornell Movie-Dialogs Corpus)

Software: Python, TensorFlow, Keras, Scikit-learn for machine learning, Flask/Django for deployment

Hardware: Local machine with GPU/Cloud computing platform (Google Colab)

**Methods:**

# Data Preparation: Tokenize and preprocess conversational data

# Model Development: Build CNN architectures to classify and generate responses

# Training: Train CNN models on conversational datasets, tuning to maximize performance

# Evaluation: Assess chatbot's responses using evaluation metrics such as BLEU

# CONCLUSION

# This project successfully developed a CNN-based chatbot capable of understanding user input and generating relevant responses. By leveraging the pattern recognition abilities of CNNs, we improved the chatbot's ability to handle diverse conversational scenarios. Future work can involve refining the chatbot's context-awareness and integrating additional deep learning models to handle more complex dialogues.

# REFERENCES

* Behera, G.: Privacy preserving c4.5 using gini index pp. 1 –4 (march 2011)
* Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984)
* Deri, L.: nprobe: an open source netflow probe for gigabit networks. In: In Proc. of Terena TNC 2003 (2003)
* Fomenkov, M., Claffy, K.: Internet measurement data management challenges. In: Workshop on Research Data Lifecycle Management. Princeton, NJ (Jul 2011)
* Grzenda, M.: Towards the reduction of data used for the classification of network flows. In: Proceedings of the 7th international conference on Hybrid Artificial Intelligent Systems - Volume Part II. pp. 68–77. HAIS’12, Springer-Verlag, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-28931-6\_7
* Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: Internet traffic classification demystified: myths, caveats, and the best practices. In: Proceedings of the 2008 ACM CoNEXT Conference. pp. 11:1–11:12. CoNEXT ’08, ACM, New York, NY, USA (2008)
* Kobiersky, P., Korenek, J., Polcak, L.: Packet header analysis and field extraction for multigigabit networks pp. 96 –101 (april 2009)
* Limwiwatkul, L., Rungsawang, A.: Distributed denial of service detection using tcp/ip header and traffic measurement analysis 1, 605 – 610 vol.1 (oct 2004)
* Moore, A., Crogan, M., Moore, A.W., Mary, Q., Zuev, D., Zuev, D., Crogan, M.L.: Discriminators for use in flow-based classification. Tech. rep. (2005)
* Ouyang, T., Ray, S., Rabinovich, M., Allman, M.: Can network characteristics detect spam effectively in a stand-alone enterprise? In: Proceedings of the 12th international conference on Passive and active measurement. pp. 92–101. PAM’11, Springer-Verlag, Berlin, Heidelberg (2011)
* Schatzmann, D., Burkhart, M., Spyropoulos, T.: Flow-level characteristics of spam and ham (291) (Aug 2008)
* Z´adn´ık, M., Michlovsk´y, Z.: Is spam visible in flow-level statistics? Tech. rep. (2009),