Alitha_Technical Analysis Document for ChatBot

(1) Chatbot Project Brief

Project Title:

In [ ]:
```
* AI-Powered Chatbot for Intelligent Assistance
```

Objective:

In [ ]:
```
* To design and develop an AI-driven chatbot capable of engaging in natural,
* human-like conversations to assist users with queries, provide information, and automate routine tasks efficiently.
```

Problem Statement:

In [ ]:
```
* Users often face delays or difficulties in accessing quick, accurate, and personalized responses.
* Manual support is time-consuming and resource-intensive.
* A chatbot can bridge this gap by delivering real-time assistance 24/7.
```

Scope:

In [ ]:
```
* Handle FAQs and repetitive queries.

* Provide personalized recommendations based on user input.

* Integrate with existing systems (website/app).

* Support text-based interactions (and potentially voice in the future).
```

Features:

In [ ]:
```
* Natural Language Understanding (NLU) to interpret user queries.

* Context awareness to maintain conversational flow.

* Multi-domain support (e.g., customer service, product inquiries, or data insights).

* Scalable architecture for future expansion.
```

Technology Stack (tentative):

In [ ]:
```
* Frontend: Web/App integration (React, Flutter, or HTML/CSS/JS)

* Backend: Python / Node.js

* AI/NLP: OpenAI GPT, spaCy, Rasa, or Dialogflow

* Database: MySQL / MongoDB / Firebase
```

Expected Outcomes:

In [ ]:
```
* Reduced support workload.

* Faster response times and improved customer satisfaction.

* Scalable foundation for future automation (voice, multilingual support).
```

Deliverables:

In [6]:
```
* Functional chatbot prototype.

* Integration with a platform (website/app).

* Documentation & user guide.
```

```
  Cell In[6], line 1
    Users often face delays or difficulties in accessing quick, accurate, and personalized responses. Manual support is time-co
nsuming and resource-intensive. A chatbot can bridge this gap by delivering real-time assistance 24/7.
          ^
SyntaxError: invalid syntax
```

(2) Types of Chatbots

a) Rule-Based Chatbots

In [ ]:
```
* Follow pre-defined scripts and keywords.
* Limited intelligence.
* Example: “If user says Hello → Reply with Hi, how can I help you?”.
```

b) Retrieval-Based Chatbots

In [ ]:
```
* Use ML/NLP to retrieve the most relevant response from a database.
* Models: TF-IDF, KNN, SVM, BERT, etc.
```

c) Generative Chatbots

In [ ]:
```
* Use deep learning (RNN, LSTM, GRU, Transformers) to generate new replies.
 Example: ChatGPT, Google Bard.
```

(3) How Does a Chatbot Work?

In [ ]:
```
* Input Processing – The chatbot reads user query.
* Natural Language Understanding (NLU) – Identifies intent (what user wants) and
  entities (medical condition, drug name, etc.).
* Response Generation – Either retrieves a pre-stored answer or generates a new
one using ML/DL.
* Output Delivery – Sends response back as text or voice.
```

a) Applications of Chatbots

In [ ]:
```
* Healthcare: Medical Q&A, symptom checking, appointment booking.
* Customer Support: Answer FAQs, troubleshoot issues.
* Banking/Finance: Balance check, transaction queries.
* Education: Student helpdesk, learning assistants.
* E-commerce: Product recommendations, order tracking.
```

(b) Benefits of Chatbots

In [ ]:
```
* 24×7 availability
* Faster response time
* Reduces human workload
* Cost-effective
* Scalable to millions of users
```

(4) Problem Statement

In [ ]:
```
* People engage in daily conversations that include greetings, casual questions, and small
talk (e.g., “Hi, how are you?”, “What school do you go to?”, “It looks like rain today”).
The challenge is to design a chatbot that can:
* Understand such casual dialogues.
* Generate or retrieve appropriate human-like responses.
* Maintain conversation flow across multiple turns.
* Simulate natural interaction similar to a human conversational partner.
* Accurately match user queries to responses using NLP/ML/DL techniques.
* Provide safe and relevant guidance instead of misleading answers.
```

(5) Motivation

In [ ]:
```
a) Human-like Interaction
* Most chatbots focus only on task-oriented queries (e.g., booking, FAQs).
* But real users also want friendly, casual conversation to feel natural.
```

```
b) User Engagement
* Small talk (“Hi, how are you?”, “Nice weather today”) helps keep users
 engaged and comfortable before moving to serious queries.

c) Foundation for Advanced Chatbots
* Learning to handle greetings and small talk is the first step toward building
more advanced conversational AI.
* Once the chatbot can handle general dialogue, it can be extended to domainspecific
tasks (e.g., medical advice, education).

d) Social Companion Role
* Such chatbots can be used as companions for lonely people, elderly users, or
students practicing English conversation.
```

(6) Dataset Description

In [ ]:

```
* Dataset Overview (dialogs.txt)
* Total dialog pairs: 3725
* User length (tokens): mean=6.35, median=6
* Bot length (tokens): mean=6.52, median=6
* Sample pairs (first 5):
   User                                              Bot
* hi, how are you doing?                           i'm fine. how about yourself?
* i'm fine. how about yourself?                    i'm pretty good. thanks for asking.
* i'm pretty good. thanks for asking.              no problem. so how have you been?
* no problem. so how have you been?                i've been great. what about you?
* i've been great. what about you?                 i've been good. i'm in school right now.
* Distributions:
```

Steps for Project:

In [ ]:

```
Step 1 — Imports, Configuration, and Seeding:
* Imports core libraries (NumPy, pandas, matplotlib, NLTK/TF/etc.) and sets random seeds for
reproducibility.
* Establishes project-wide constants (paths, hyperparameters) used later.
* Expectation: No output besides library versions or seed confirmation logs.
```

```
Step 2 — Load dialogs.txt and Quick Peek:
* Reads the tab-separated file `dialogs.txt` into a DataFrame with columns: `user`, `bot`.
* Prints dataset size and previews head rows to verify delimiter and encoding.
* Observation: We verified 3,725 pairs and show the first few rows above.
* Observation: Data successfully parsed with two columns (user, bot). See sample table above.

Step 3 — Text Cleaning (lowercase, punctuation/HTML removal):
* Defines helper functions to remove HTML tags, punctuation, lowercase text, and normalize
whitespace.
* Produces `user_clean` and `bot_clean` columns for downstream modeling.
* Observation: Cleaning reduces noise (e.g., punctuation) and standardizes casing; token counts
typically shrink slightly.

Step 4 — Visualization:
* Plots utterance length histograms and top word frequencies to understand distribution and
vocabulary.
* Observation: See embedded histograms and top-20 token bar chart; most utterances are short
(under ~10 tokens).
* Embedded visualizations generated from your dataset are shown earlier in the report.

Step 5 — Train/Test Split:
* Splits the dataset into training and testing subsets, usually stratified or random shuffling with
fixed seeds.
* Observation: Expect printed sizes of train/test; typical split is 80/20 or similar.
* Observation: With fixed seed, split is reproducible. Expect similar class/length distributions in
both splits.

Step 6 — Retrieval: TF-IDF + Cosine Similarity:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

Step 7 — Retrieval: TF-IDF + KNN Classification:
* Uses TF-IDF vectors with K-Nearest Neighbors classification to predict most likely response
class/cluster.
* Observation: Reports classification accuracy; K is tuned to balance bias/variance.

Step 8 — Retrieval: Linear SVM Classification (+ Top-3):
* Trains a linear SVM over TF-IDF features to classify or rank responses (often stronger than
KNN on sparse text).
* Observation: Expect precision/recall/accuracy metrics and possibly Top-3 predictions.
```

```
Step 9 — (Optional) SBERT Semantic Retrieval:
* (Optional) Uses Sentence-BERT embeddings to compute semantic similarity, improving
retrieval beyond word overlap.
* Observation: Better matching for paraphrases; requires pre-trained transformer models.

Step 10 — Deep Learning: Tokenization and RNN/LSTM/GRU Classifiers:
* Converts text to integer sequences and pads to max length. Builds deep models
(RNN/LSTM/GRU) over embeddings.
* Observation: Model summaries show embedding and recurrent layers; training logs show
loss/accuracy per epoch.

Using Bi_Directional:
* Applies a Bidirectional wrapper to RNN/LSTM to capture both past and future context in
sequences.
* Observation: Often improves accuracy at the cost of more parameters.

Improving Accuracy:
* Hyperparameter tuning (units, learning rate, dropout), architecture tweaks, or data
augmentation to raise accuracy.
* Observation: Expect incremental gains; track via validation accuracy curves.

Bi-Directional:
* Applies a Bidirectional wrapper to RNN/LSTM to capture both past and future context in
sequences.
* Observation: Often improves accuracy at the cost of more parameters.

Step 11 — BLEU Evaluation (text quality proxy):
* Computes BLEU, a machine translation style metric, to evaluate generated text quality vs.
references.
* Observation: BLEU provides corpus-level quality proxy; interpret with caution for dialog tasks.

Step 12 — Consolidated Report:
* Collates metrics (accuracy, BLEU), confusion matrices, and example outputs into a final
summary.
* Observation: Presents overall performance comparison among approaches.

Creating Final Chartbot based on Model RNN, because it is Proving
Maximum Accuracy of 67%:
* Chooses the best-performing deep model (RNN-based) to build the final chatbot pipeline.
* Observation: The notebook notes an accuracy of ~67% for the chosen model (as reported in
```

```
your notes).
* Observation: As per your notebook notes, final selected model achieved ~67% accuracy.
No code cells under this section in the notebook.
Explanation:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

First Approach:
* Baseline or initial approach pipeline (likely TF-IDF retrieval/classification), serving as a
performance benchmark.
* Observation: Establishes reference accuracy and qualitative examples.

Second Approach:
* Enhanced approach (e.g., deep learning or semantic retrieval) improving upon the first baseline.
* Observation: Compares metrics against the first approach to demonstrate gains.

3rd Approach:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

Tokenization & sequence preparation:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

Build Sequenvce to Sequence Models:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

Training:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

Build inference models:
* This section contains supporting code and analysis as part of the overall chatbot pipeline.
* Observation: The snippet above; outputs depend on execution context.

chat:
* This section contains supporting code and analysis as part of the overall chatbot pipeline
```