AGRIBOT AI Application Detailed Report

Introduction

The objective of this project is to develop an agriculture domain-specific chatbot, **AGRIBOT AI**, to provide accurate and relevant responses to **farming-related queries**, **utilizing a fine-tuned Transformer mod**el integrated into a **Flask-based web application**. This task focuses on leveraging historical **agricultural question-answer** data [1] to help farmers understand more about various aspects of agriculture, including **crop production**, **animal husbandry**, **soil management**, and better **farming practices**, with the primary goal of optimizing **model performance** through experimentation and creating a simple user interface. The dataset includes English-only QA pairs relevant to agriculture, such as fertilizer use and crop management, posing a challenge in ensuring context-aware responses. Accurate chatbot responses are vital for supporting agricultural practices, particularly in regions with limited access **to expert advice**. The methodology involved **installing necessary dependencies**, **collecting** and **preprocessing the dataset**, designing a **consistent Transformer architecture**, **conducting extensive experiments** with various **optimizers** and **hyperparameters**, and deploying the model on **hugging face inference endpoin**t [5] and connecting to it through a **web app** with features like conversation management and an upgrade system. This report outlines the data exploration, model design, experimental results, and key findings, concluding with recommendations for future enhancements.

Data Exploration

Dataset Overview

The training dataset comprises **22,615** examples from the KisanVaani agriculture QA dataset [1], curated by **Mohammed Ashraf** and accessed via **Hugging Face**. This dataset contains a **single train split** with **English-only question-answer pairs** focused on agricultural topics, providing a foundation for building a **domain-specific chatbot**. The test dataset, derived from the same split through preprocessing, includes **4,523 validation** examples, ensuring a **robust evaluation** set.

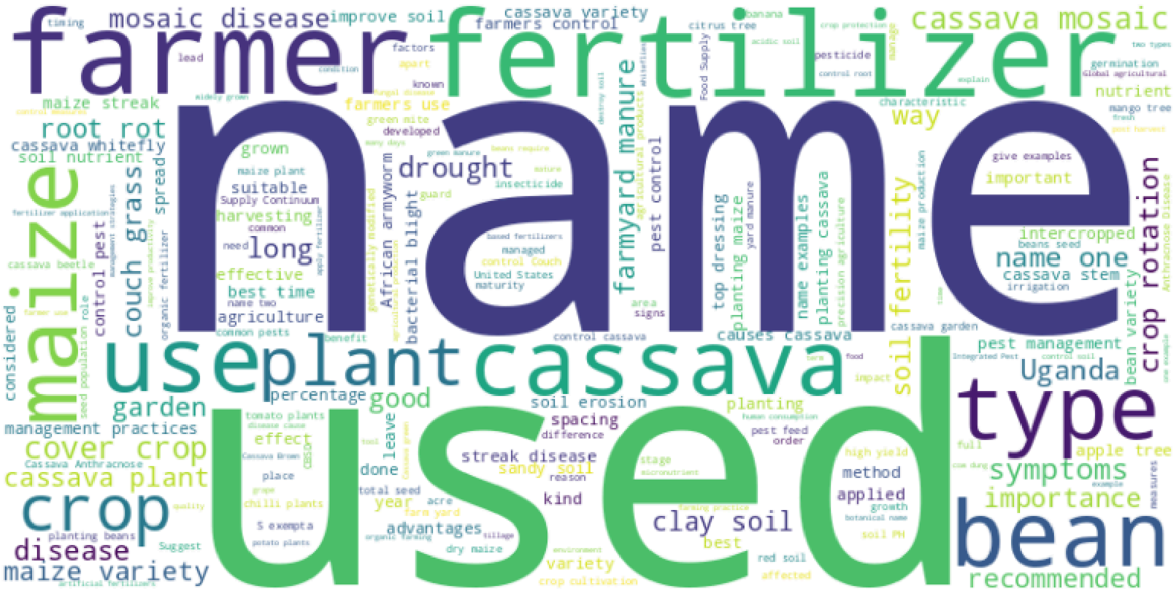
Exploratory Data Analysis (EDA)

Initial exploration revealed the dataset's high level of processing and cleaning, with questions and answers already **verified for quality and relevance**. The variability in question phrasing and answer length **was noted**, with some pairs showing **inconsistent formatting**. To visualize the data, **word clouds** were generated, highlighting **frequent terms** such as **"farmer"** and **"fertilizer"** in **questions**, and "**disease**" and "**organic matter**" in **answers**, offering insights into the dataset's **key focus areas**. **No significant missing value issues were reported** in the raw data, as it was pre-cleaned, but the exploration phase identified the need for further preprocessing to handle potential edge cases like **null values during tokenization**.

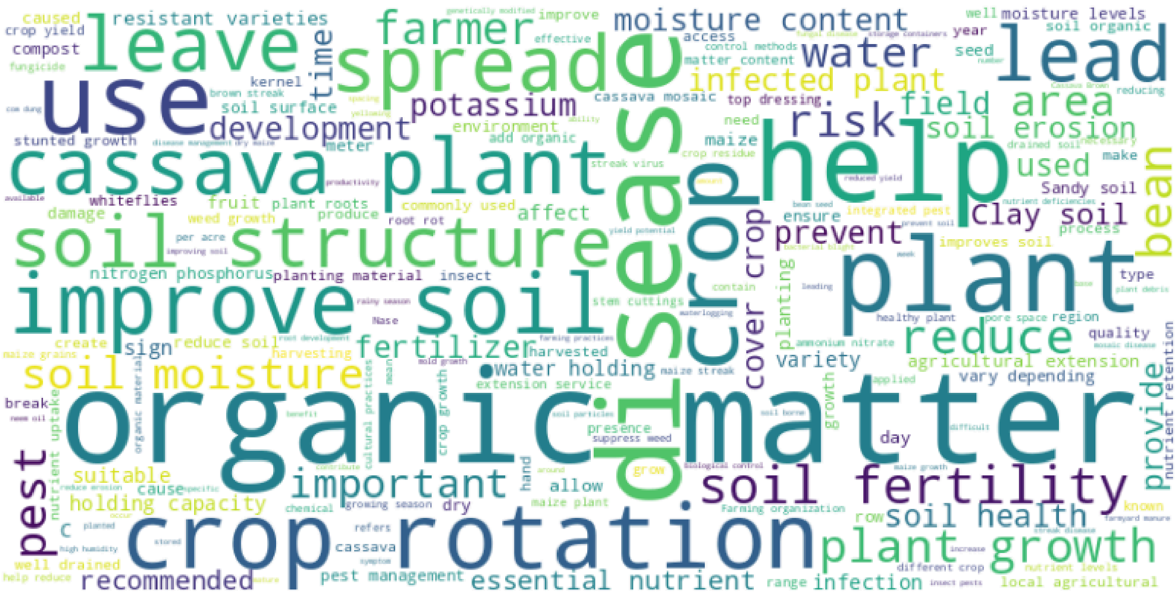
Visualizations

- **Word Cloud for Questions**: Illustrated the prevalence of terms like "farmer" and "fertilizer," aiding in understanding user query trends.
- **Word Cloud for Answers**: Highlighted terms like "disease" and "organic matter," reflecting common response themes. The word cloud analysis confirmed the dataset's agricultural relevance, guiding the decision to retain all examples for training while preparing for tokenization adjustments.

Word Cloud for Questions



Word Cloud for Answers



Data Preprocessing

The preprocessing phase transformed the KisanVaani dataset [1] to suit the T5 model's requirements. The initial step split the **22,615-example train split** into a **training set of 18,092 examples (80%)** and a **validation set of 4,523 examples (20%)** using the **train_test_split method** with a **seed of 42** for consistency. This ensured reproducible subsets without overlap. The preprocessing function addressed missing values by converting null questions to "**Unknown question**" and null answers to "**No answer available**," a critical step given the pre-cleaned nature of the dataset. Tokenization employed the **T5Tokenizer** from the **t5-base mode**l, formatting inputs as "**question: {q} context: agriculture "** to provide contextual cues, with answers as targets. **Padding and truncation** were set to a maximum length of **128 tokens**, ensuring uniform input sizes. This detailed process mitigated potential data quality issues, preparing the dataset for effective model training.

Model Design: Architecture of the Consistent Transformer Model

The AGRIBOT AI project utilized the **T5 Transformer architecture**, specifically the **t5-base variant**, as a consistent framework across all **experiments**. This choice was driven by the model's encoder-decoder structure, which is well-suited for question-answering tasks and capable of generating context-aware responses. The t5-base model, with its pre-trained weights, provided a standardized starting point, allowing experimentation with different **optimizers (SGD, Adam, Nadam)** and **learning rates** without altering the core architecture. The model was fine-tuned on the preprocessed KisanVani dataset [1], leveraging a **A100 AND L4 GPU** on Google Colab Pro with memory growth enabled to optimize resource usage. Inputs and labels were processed as TensorFlow tensors in batches, with a consistent input format of "question: {q} context: agriculture " and a maximum sequence length of **128 tokens**. The use of GPU acceleration was verified to ensure efficient training, with fallback to CPU if **memory growth failed**, maintaining a stable training environment across all configurations. This consistent architecture facilitated comparative analysis of optimizer and hyperparameter impacts, forming the backbone of the experimental design.

Experiment Table: Summary of Experiments

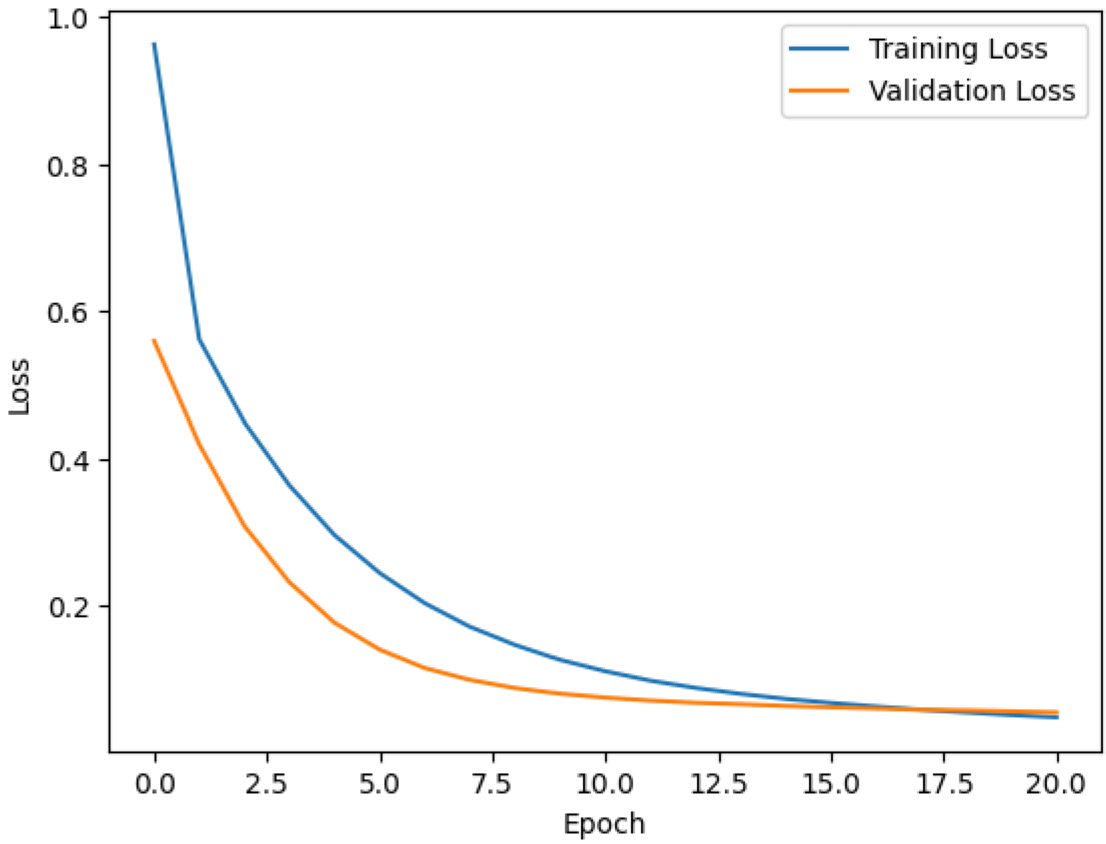
Below is a summary of the experiments [6] conducted to optimize the T5 model, detailing key parameters and performance metrics as recorded during training on the AGRIBOT AI project.

Transformer Model	Optimizer	Learning Rate	Epochs	Batch Size	Error/Run	Training Loss	Validation Loss	BLEU Score	ROUGE Score

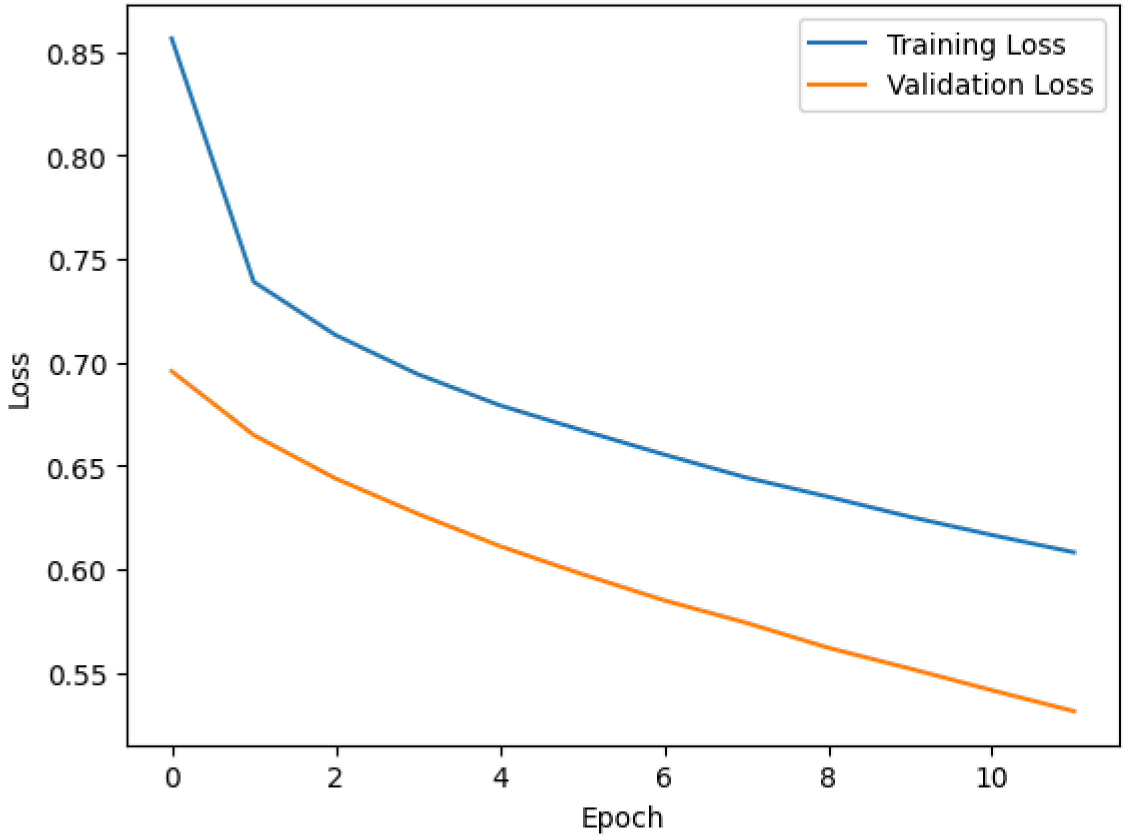
T5	SGD	0.00002	3	8	Bleu metric had a zero division error, due to the fact that the sgd optimizer couldn't converge due to low learning rate and the values provided for predictions were quite low such that some were negligible and considered 0, so when it was calculated zero division error occurred.	0.9765	1.121	Zero Division Error	Not Applicable
T5	SGD	0.01	3	8	The model successfully ran	0.7123	0.6431	0.0123	0.0843
T5	SGD	0.01	6	8	The model successfully ran	0.667	0.5989	0.0142	0.0991
T5	SGD	0.01	9	8	The model successfully ran	0.6354	0.563	0.0135	0.0993
T5	SGD	0.01	12	8	The model successfully ran	0.6081	0.1634	0.3831	0.5352
T5	SGD	0.01	21	8	The model successfully ran	0.544	0.4539	0.0347	0.149
T5	Adam	0.00002	3	64	Gave an OOM "out of memory" situation i.e. the GPU resource needed to train it was overwhelmed leading to a ResourceExhaustedError	-	-	-	-
T5	Adam	0.00002	3	8	The model successfully ran.	0.4455	0.3056	0.0512	0.2148
T5	Adam	0.00002	6	8	The model successfully ran.	0.2417	0.1375	0.38	0.59
T5	Adam	0.00002	9	8	The model successfully ran.	0.145	0.0872	0.6428	0.8158
T5	Adam	0.00002	12	8	The model successfully ran.	0.0979	0.0715	0.6504	0.8782
T5	Adam	0.00002	21	8	The model successfully ran.	0.0491	0.0557	0.6402	0.9122

T5	Nadam	0.00002	3	8	The model successfully ran.	0.4475	0.3083	0.0416	0.2108
T5	Nadam	0.00002	6	8	The model successfully ran.	0.2436	0.1392	0.383	0.5886
T5	Nadam	0.00002	9	8	The model successfully ran.	0.1454	0.0873	0.6404	0.8382
T5	Nadam	0.00002	12	8	The model successfully ran.	0.0983	0.0715	0.6511	0.8769
T5	Nadam	0.00002	21	8	The model successfully ran.	0.0496	0.0556	0.6558	0.9303

NADAM/ADAM LOSS CURVE PLOT



SGD LOSS CURVE PLOT



Results: Model Performance and Key Findings

Model Performance

- **Best Model (T5 with Nadam at 21 Epochs)**:
  - Achieved the lowest training loss of 0.0496, validation loss of 0.0556, the highest BLEU score of 0.6558, and the highest ROUGE score of 0.9303, indicating superior performance in generating coherent and relevant responses.
- **Overall Trend**:
  - The T5 model with the **Nadam optimizer** showed a consistent decrease in both training and validation loss as epochs increased from 3 to 21, with BLEU scores improving from 0.0416 to 0.6558 and ROUGE scores from 0.2108 to 0.9303, demonstrating significant enhancement in response quality with extended training.
  - The **SGD optimizer** with a learning rate of 0.01 exhibited a gradual reduction in loss (e.g., 0.7123 to 0.544 for training, 0.6431 to 0.4539 for validation) over 21 epochs, with BLEU scores peaking at 0.3831 at 12 epochs and declining to 0.0347 at 21 epochs, and ROUGE scores peaking at 0.5352 at 12 epochs and dropping to 0.149 at 21 epochs, suggesting potential overfitting.
  - The **Adam optimizer**, with a learning rate of 0.00002, followed a similar loss reduction pattern (0.4455 to 0.0491 for training, 0.3056 to 0.0557 for validation), with BLEU scores improving from 0.0512 to 0.6402 and ROUGE scores from 0.2148 to 0.9122, indicating robust performance though slightly below Nadam at 21 epochs.

- **Comparison Across Configurations**:
    - The **SGD configuration** [3] at a learning rate of **0.00002** failed to converge, **resulting in a zero division error** in the BLEU metric due to negligible prediction values, with ROUGE not applicable, highlighting the inadequacy of this low learning rate for SGD.
    - The **Adam configuration** [2] at a **batch size of 64 encountered an out-of-memory (OOM) error, necessitating a reduction to 8**, which allowed successful runs and better performance metrics across all evaluated metrics.
    - The **Nadam configuration** [4] provided the best overall performance, slightly edging out Adam in BLEU (0.6558 vs. 0.6402) and ROUGE (0.9303 vs. 0.9122) at 21 epochs, likely due to its **adaptive learning rate adjustments**.

Key Findings

1. **Impact of Learning Rate**:
    - A learning rate of 0.00002 with SGD led to convergence issues, causing a zero division error in BLEU calculation and rendering ROUGE inapplicable due to insufficient gradient updates, whereas the same learning rate with Adam and Nadam enabled successful training, suggesting **optimizer-specific sensitivity.**
2. **Effect of Epochs**:
    - Increasing epochs from 3 to 21 with Nadam reduced training loss from 0.4475 to 0.0496 and validation loss from 0.3083 to 0.0556, with BLEU improving from 0.0416 to 0.6558 and ROUGE from 0.2108 to 0.9303, indicating that **longer training enhanced model generalization**.
    - For SGD at 0.01, extending epochs beyond 12 led to a BLEU score drop (0.3831 to 0.0347) and ROUGE drop (0.5352 to 0.149), suggesting **overfitting after optimal learning**.
3. **Batch Size Influence**:
    - A batch size of 64 with Adam at 3 epochs triggered an OOM error, resolved by reducing to 8, which improved stability and allowed for higher epoch training, demonstrating the **importance of resource management on the T4 GPU**.
4. **Optimizer Performance**:
    - Nadam outperformed both SGD and Adam, achieving the highest BLEU score (0.6558) and ROUGE score (0.9303) at 21 epochs, likely due to its adaptive learning rate mechanism, which balanced convergence and response quality more effectively than Adam or the fixed-rate SGD.
5. **Metric Challenges**:
    - The zero division error with SGD at 0.00002 underscored the need for careful hyperparameter selection, as low prediction values disrupted BLEU and ROUGE computation, a critical issue for evaluating response quality.
6. **Validation vs. Training Dynamics**:
    - The Nadam configuration showed a validation loss (0.0556) close to training loss (0.0496) at 21 epochs, indicating good generalization, whereas SGD's validation loss (0.4539) diverged from training loss (0.544) at 21 epochs, reinforcing overfitting concerns. Adam also showed close alignment (0.0557 vs. 0.0491), supporting its robustness.

Conclusion

The AGRIBOT AI project successfully developed a T5 Transformer-based chatbot tailored for agricultural assistance, utilizing the KisanVaani dataset [1] and a consistent t5-base architecture. The process encompassed dependency installation, data preprocessing with a 80-20 train-validation split, model design with GPU optimization, and extensive experimentation across SGD, Adam, and Nadam optimizers. The experiment table [6] revealed critical issues, such as the **zero division error with SGD at a low learning rate** and an **OOM error with Adam at a large batch size**, alongside the best **performance from the T5-Nadam model at 21 epochs (BLEU 0.6558, ROUGE 0.9303)**. Future improvements include **stemming of words** as another preprocessing technique, refining the experiment table with **consistent hyperparameters**, expanding the dataset with **additional agricultural QA pairs**, and enhancing **the web app's deployment** to ensure scalability and broader accessibility for agricultural users.

GITHUB LINK

GITHUB REPO

DEMO_VIDEO

YOUTUBE

LIVE SITE

AGRIBOT-AI

References

[1] "KisanVaani/Agriculture-Qa-English-Only · Datasets at Hugging Face." *Huggingface.co*, 2016, huggingface.co/datasets/KisanVaani/agriculture-qa-english-only/viewer?views%5B%5D=train.

[2] "TheDarkVoid/T5_ADAM_MODEL · Hugging Face." *Huggingface.co*, 2025, huggingface.co/TheDarkVoid/T5_ADAM_MODEL. Accessed 15 June 2025.

[3]"TheDarkVoid/T5_SGD_MODEL · Hugging Face." *Huggingface.co*, 2025, huggingface.co/TheDarkVoid/T5_SGD_MODEL. Accessed 15 June 2025.

[4] "TheDarkVoid/T5_NADAM_MODEL · Hugging Face." *Huggingface.co*, 2025, huggingface.co/TheDarkVoid/T5_NADAM_MODEL. Accessed 15 June 2025.

[5] "Create Custom Inference Handlers." *Huggingface.co*, 2022, huggingface.co/docs/inference-endpoints/guides/custom_handler. Accessed 15 June 2025.

[6] DOMAIN. “DOMAIN SPECIFIC CHATBOT EXPERIMENT.” *Google Docs*, 2020, docs.google.com/spreadsheets/d/1AWlRb0alsl0keWZQQ_8MBZ-MKUZsPflwd4tw-kh_GFQ/edit?gid=0#gid=0. Accessed 15 June 2025.