



LSPU STUDENT AI-DE

CHATBOT

MEMBERS



ANGELES, KHATRINA



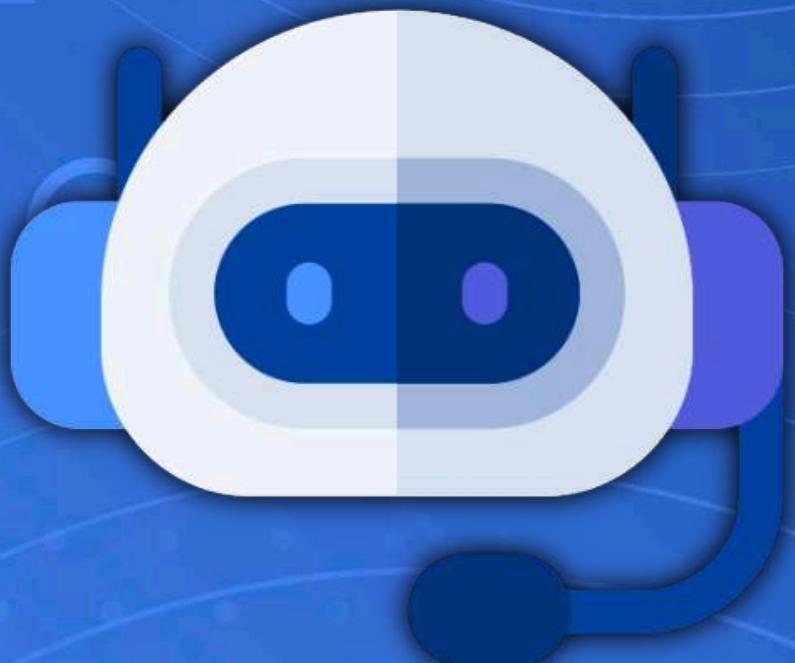
CORONADO, NIXON



SOFER, ERL TEODEMAR

INTRODUCTION

INTRODUCTION



The LSPU Student AI-de is a chatbot that aims to be the FAQ version of the university student handbook, able to answer questions about university policies, available college courses, a guide to enrollment, and navigation of the university system.

INTRODUCTION

DOMAIN

The scope of the chatbot includes all students enrolled or to be enrolled, and staff at the Laguna State Polytechnic University.

REASONING

We chose an instruction-based method as required by the instructor for more dynamic responses. We thought it was appropriate with a dataset as large as the student handbook because the machine learning model can generate responses based from the patterns in the training data.



INTRODUCTION

MACHINE LEARNING MODEL

Our chatbot utilizes the transformers-base model T5-base (Text-to-Text Transfer Transformer) for the natural language processing and the response generation.

T5 is not limited to a specific task, as it can handle data preparation, tokenization, training, response generation, and evaluation metrics, which is ideal for our chatbot,



DATA PREPARATION

DATA PREPARATION

DATASET SAMPLE

```
{  
    "question": "What graduate programs are offered by the College of Teacher Education?",  
    "answer": "The College of Teacher Education offers the following programs: Doctor of Education (with majors in Educational Management and specializations in Tertiary, Secondary, and Basic Education), Doctor o  
},  
{  
    "question": "What graduate program is offered by the College of Law?",  
    "answer": "The College of Law offers the Juris Doctor program at the Sta Cruz Campus."  
},  
{  
    "question": "What are the classifications of students based on admission?",  
    "answer": "The classifications of students based on admission include: new student, transferee student, cross-enrollee student, foreign student, and returnee student."  
},  
{  
    "question": "What are the classifications of students based on curriculum level?",  
    "answer": "The classifications of students based on curriculum level include: freshman, sophomore, junior, and senior."  
},  
{  
    "question": "What are the classifications of students based on placement?",  
    "answer": "The classifications of students based on placement include: full-time regular and irregular."  
},  
{  
    "question": "What are the admission requirements for LSPU?",  
    "answer": "Admission requirements include passing the LSPU entrance examination, meeting institutional qualifications, and adhering to non-discriminatory admission policies."  
},  
{  
    "question": "What is the policy on free higher education at LSPU?",  
    "answer": "All students currently enrolled at the time of the affectivity of the Universal Access to Quality Tertiary Education Act of 2017 shall be exempt from paying tuition and other school fees for units  
},  
{  
    "question": "What are the guidelines for foreign students' admission?",  
    "answer": "International students may be admitted to the College of their choice in accordance with guidelines set by the Philippine Bureau of Immigration, Commission on Higher Education, and the University,  
},  
{  
    "question": "What are the English language requirements for foreign students?",  
    "answer": "Foreign student applicants who are not native English speakers must have a minimum score of 500 in the Test of English as Foreign Language (TOEFL) or complete an Intensive English Language Course."  
},  
{  
    "question": "Which campus offers the Bachelor of Science in Nursing?",  
    "answer": "The Bachelor of Science in Nursing is offered at the College of Nursing and Allied Health, Sta Cruz Campus."  
},
```

DATA PREPARATION

DATASET DESCRIPTION

Using a question-and-answer format allows for straightforward mapping during training. The model learns to associate specific questions with their corresponding answers, making it easier to evaluate its performance based on how accurately it predicts answers for given questions.

Asking questions clearly helps the model generate more coherent and contextually appropriate responses, making it a more effective conversational tool.



DATA PREPARATION

PREPROCESSING STEPS

- **Load Data:** The `load_and_preprocess_data` function opens and reads a JSON file containing question-answer pairs.
- **Iterate Through Data:** For each item in the loaded data
 - Append the original question-answer pair to `augmented_data`.
 - Extract the question and answer from the current item.
- **Return Augmented Data:** The function returns the list of augmented question-answer pairs.
- **Create Dataset:** The `create_dataset` function takes the augmented data as input. Extracts questions and answers into separate lists.
 - Splits the data into training, validation, and test sets using `train_test_split` from scikit-learn, ensuring that 80% of the data is for training and 20% is split evenly for validation and testing.
- **Return Structured Datasets:** The function returns a dictionary containing three datasets (train, validation, test) formatted for use in model training.

```
def load_and_preprocess_data(file_path):
    with open(file_path, 'r') as f:
        data = json.load(f)

    augmented_data = []
    for item in data:
        augmented_data.append(item)

        question = item['question']
        answer = item['answer']

        if question.lower().startswith('what is'):
            new_question = 'Could you tell me' + question[7:]
            augmented_data.append({'question': new_question, 'answer': answer})

        if not question.lower().startswith('can you'):
            new_question = f"Can you explain {question.lower()}"
            augmented_data.append({'question': new_question, 'answer': answer})

    return augmented_data

def create_dataset(data):
    questions = [item['question'] for item in data]
    answers = [item['answer'] for item in data]

    train_questions, test_questions, train_answers, test_answers = train_test_split(
        questions, answers, test_size=0.2, random_state=42
    )

    test_questions, val_questions, test_answers, val_answers = train_test_split(
        test_questions, test_answers, test_size=0.5, random_state=42
    )

    return {
        'train': Dataset.from_dict({'question': train_questions, 'answer': train_answers}),
        'test': Dataset.from_dict({'question': test_questions, 'answer': test_answers}),
        'validation': Dataset.from_dict({'question': val_questions, 'answer': val_answers})
    }
```

DATA PREPARATION

- **Tokenization:** The preprocess_function tokenizes the questions and answers using the T5 tokenizer. It converts questions into a format suitable for the T5 model by adding "answer question:" as a prefix.
- **Numerical Conversion:** The tokenizer converts text into numerical IDs that the model can understand, pads sequences to a fixed length, and truncates longer sequences.
- **Input and Label Preparation:** Finally, the function prepares the inputs and labels for the model by associating each input question with its corresponding target answer. This involves adding the labels (target answers) to the model inputs, ensuring that both inputs and outputs are correctly formatted for training.

```
def preprocess_function(examples, tokenizer, max_length=384):
    inputs = [f"answer question: {q.strip()}" for q in examples['question']]
    targets = [a.strip() for a in examples['answer']]

    model_inputs = tokenizer(
        inputs,
        max_length=max_length,
        padding='max_length',
        truncation=True,
        return_tensors="pt"
    )

    labels = tokenizer(
        targets,
        max_length=max_length,
        padding='max_length',
        truncation=True,
        return_tensors="pt"
    )

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

MODEL IMPLEMENTATION

MODEL IMPLEMENTATION

REASONING TECHNIQUE

The reasoning technique is primarily instruction-based reasoning through the application of T5-base (Text-to-Text Transfer Transformer) model. This approach involves treating all natural language processing tasks as text-to-text problems, which means it can create responses based on what you input.

The use of T5-base as opposed to T5-small (aside from T5-large which Google Colab could not handle) is it has a larger number of parameters, which allows it to capture more complex patterns and nuances in the data. This extra capacity generally results in better understanding and generating responses, especially for tasks that need contextual awareness and intricate language processing.



MODEL IMPLEMENTATION

The core of the chatbot lies in the Chatbot class, which handles:

- Model Initialization: Loading the pre-trained T5 model and tokenizer.
- Question Preprocessing: Cleaning and formatting the user's question.
- Response Generation: Encoding the question, generating responses using the T5 model, and evaluating confidence scores.



MODEL IMPLEMENTATION

- **Model Initialization:** The Chatbot class initializes the chatbot by loading the trained model and tokenizer. It sets the device for computation (GPU if available, else CPU) and sets the model to evaluation mode. It defines a fallback response for when the model's confidence is low and includes predefined responses for common greetings.

```
class Chatbot:  
    def __init__(self, model_path):  
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")  
        self.tokenizer = T5Tokenizer.from_pretrained(model_path)  
        self.model = T5ForConditionalGeneration.from_pretrained(model_path).to(self.device)  
        self.model.eval()  
        self.fallback_response = "I'm sorry, I don't have information on that. Can you please rephrase or ask a different question?"  
        self.predefined_responses = {  
            "hi": "Hello! How can I assist you today?",  
            "hello": "Hello! How can I assist you today?",  
            "hey": "Hello! How can I assist you today?",  
            "thanks": "You're welcome! Feel free to ask more questions.",  
            "thank you": "You're welcome! Feel free to ask more questions.",  
            "bye": "Goodbye! Have a great day!",  
            "goodbye": "Goodbye! Have a great day!",  
            "thankyou": "You're welcome! Feel free to ask more questions."  
        }
```

MODEL IMPLEMENTATION

- **Question Preprocessing:** This function converts the input question to lowercase, removes unwanted characters using regular expressions, and ensures it ends with a question mark. This preprocessing helps standardize user input for better model understanding.

```
def preprocess_question(self, question):  
    question = question.lower().strip()  
    question = re.sub(r'[^a-zA-Z\s?.!]', '', question)  
    if not question.endswith('?'):  
        question += '?'  
    return question
```

MODEL IMPLEMENTATION

- **Response Generation:** This function generates a response based on the user's question. It first checks for predefined responses. If not found, it preprocesses the question, encodes it using the tokenizer, and generates a response using the model. The confidence score is calculated, and if it's below the threshold, the fallback response is returned.

```
def generate_response(self, question, max_length=256, num_beams=5, confidence_threshold=0.5):
    if question.lower() in self.predefined_responses:
        return self.predefined_responses[question.lower()]

    processed_question = self.preprocess_question(question)
    input_text = f"answer question: {processed_question}"

    inputs = self.tokenizer(
        input_text,
        return_tensors="pt",
        max_length=max_length,
        truncation=True,
        padding=True
    ).to(self.device)

    with torch.no_grad():
        outputs = self.model.generate(
            inputs["input_ids"],
            max_length=max_length,
            num_beams=num_beams,
            length_penalty=1.0,
            early_stopping=True,
            no_repeat_ngram_size=2,
            temperature=0.7,
            output_scores=True,
            return_dict_in_generate=True
        )

        logits = outputs.scores[0]
        probs = torch.softmax(logits, dim=-1)
        confidence = probs.max().item()

        response = self.tokenizer.decode(outputs.sequences[0], skip_special_tokens=True)

        if confidence < confidence_threshold:
            return self.fallback_response

    return response
```

TRAINING THE MODEL

TRAINING THE MODEL

The ***train_model*** function loads the pre-trained T5 tokenizer and model. It then preprocesses the datasets using the defined ***preprocess_function*** and tokenizes them.

```
def train_model(datasets, model_name="t5-base", output_dir="../results"):
    tokenizer = T5Tokenizer.from_pretrained(model_name)
    model = T5ForConditionalGeneration.from_pretrained(model_name)

    tokenized_datasets = {
        split: datasets[split].map(
            lambda x: preprocess_function(x, tokenizer),
            batched=True,
            remove_columns=datasets[split].column_names
        )
        for split in datasets.keys()
    }
```

TRAINING THE MODEL

The **TrainingArguments** class sets up the training parameters, such as the output directory, learning rate, batch size, and number of epochs.

The learning rate is like a speed limit for how fast the model learns. It decides how much to change the model after each mistake. We set it at $5e-5$ because it's balanced in terms of speed and accuracy. We initially used $1e-5$, but it was too slow, and the results were inaccurate.

The batch size is how many examples the model looks at before updating its settings. A batch size of 6 for the training and evaluation batch was also the limit we could do in Google Colab.

An epoch is one complete run through the entire training dataset. In each epoch, the model learns from every question and answer, helping it improve. 17 epochs was around the limit we could do in Google Colab.

```
training_args = TrainingArguments(  
    output_dir=output_dir,  
    evaluation_strategy="steps",  
    eval_steps=50,  
    learning_rate=5e-5,  
    per_device_train_batch_size=6,  
    per_device_eval_batch_size=6,  
    num_train_epochs=17,  
    weight_decay=0.01,  
    save_total_limit=2,  
    logging_steps=10,  
    load_best_model_at_end=True,  
    metric_for_best_model="eval_loss",  
    greater_is_better=False,  
    gradient_accumulation_steps=2,  
    warmup_steps=100,  
    fp16=True  
)
```

TRAINING THE MODEL

The custom trainer is initialized with the model, training arguments, datasets, and tokenizer.

This trainer is configured with the model, training arguments, tokenized datasets for training and evaluation, and an early stopping callback to prevent overfitting.

```
class CustomTrainer(Trainer):
    def compute_metrics(self, eval_pred):
        predictions, labels = eval_pred

        decoded_preds = self.tokenizer.batch_decode(predictions, skip_special_tokens=True)
        decoded_labels = self.tokenizer.batch_decode(labels, skip_special_tokens=True)

        binary_preds = [1 if p == 1 else 0 for p, l in zip(decoded_preds, decoded_labels)]
        binary_labels = [1] * len(decoded_labels)

        return {
            'accuracy': accuracy_score(binary_labels, binary_preds),
            'f1': f1_score(binary_labels, binary_preds, average='binary'),
            'recall': recall_score(binary_labels, binary_preds, average='binary'),
            'precision': precision_score(binary_labels, binary_preds, average='binary')
        }
```

```
trainer = CustomTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['validation'],
    tokenizer=tokenizer,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)],
)
```

TRAINING THE MODEL

The trainer's train method is called to start the training process.
The function returns the trained model, tokenizer, and trainer
instance for further use or evaluation.

```
trainer.train()
```

```
return model, tokenizer, trainer
```

TRAINING THE MODEL

The training loss measures the model's fit to the training data. If the value is decreasing, then the model is learning effectively. As we can see here in the training output, it starts at 7.092300 at Step 50, but continually drops down more or less to 0.29500 at Step 2000.

The validation loss reflects how well the model performs on new unseen data. It's also under the logic of when it decreases, it's generalizing, not overfitting. We can also see that it performed effectively as it started at a value of 0.56465 and ended at around 141067.

Step	Training Loss	Validation Loss
50	7.092300	0.564365
100	0.654700	0.309032
150	0.573300	0.258273
200	0.485000	0.242414
250	0.457100	0.233813
300	0.452700	0.226121
350	0.475900	0.219767
400	0.471100	0.214069
450	0.474900	0.208969
500	0.459200	0.204147
550	0.444700	0.199451
600	0.382000	0.194656
650	0.437500	0.190209
700	0.477600	0.186207
750	0.386700	0.183146
800	0.362700	0.179388
850	0.348300	0.175868
900	0.415500	0.172316
950	0.301300	0.169516
1000	0.310600	0.166500
1050	0.314700	0.163877
1100	0.293600	0.161523
1150	0.313700	0.158917
1200	0.344500	0.156927
1250	0.280400	0.154951
1300	0.262500	0.152794
1350	0.292200	0.150751
1400	0.335500	0.149414
1450	0.322600	0.147912
1500	0.259400	0.146629
1550	0.323400	0.145530
1600	0.297800	0.144499
1650	0.273100	0.143778
1700	0.297700	0.143060
1750	0.293000	0.142536
1800	0.260500	0.142005
1850	0.280700	0.141686
1900	0.241000	0.141343
1950	0.280000	0.141100
2000	0.292500	0.141067

TESTING AND EVALUATION

TESTING AND EVALUATION

CHATBOT EXECUTION

```
... LSPU Student Student AI-de
```

```
=====
```

```
Initializing chatbot...
```

```
LSPU Student AI-de is ready! Type 'exit' or 'quit' to stop.  
Type 'metrics' to see the model's performance metrics again.
```

```
You: hello
```

```
Bot: Hello! How can I assist you today?
```

```
You: is wearing uniforms required in LSPU?
```

```
Bot: Wearing uniforms is required for all students, with exceptions for students with disabilities.
```

```
You: what is the procedure for dropping a course?
```

```
Bot: Students must submit a written request to the Dean of the College before dropping a course.
```

```
You: What is the policy regarding clothing and gender representation at LSPU?
```

```
Bot: LSPU enforces dress codes that allow transgender students to dress in accordance with their gender identity while adhering to university rules and regulations.
```

```
You: What should a student do if they have concerns about their enrollment status?
```

```
Bot: A student should contact the Office of the Registrar if they have concerns about their enrollment status.
```

```
You: what is the main campus of LSPU?
```

```
Bot: The main campus is LSPU-Sta. Cruz Campus (Main Campus).
```

TESTING AND EVALUATION

CHATBOT EXECUTION

You: is smoking allowed in LSPU?

Bot: Smoking within the University premises is strictly prohibited.

You: is drinking alcoholic beverages allowed in LSPU?

Bot: Drinking alcoholic beverages is prohibited.

You: does the LSPU offer counseling services?

Bot: The LSPU offers counseling services for students, faculty, and staff.

You: how can students access their academic records?

Bot: Students can access their academic records by logging into the Student Records Portal.

You: what is the student development services at LSPU?

Bot: Student development services are provided by the Office of the Registrar to facilitate communication between students, faculty, and the university administration.

You: wWhich campus offers the bachelor of science in hospitality management?

Bot: The Bachelor of Science in Hospitality Management is offered at the College of Hotel Management and Tourism (CHMT), All Campuses.

You: what are academic awards in LSPU?

Bot: Academic awards are given to students who excel academically and actively participate in their field of study.

You: what is student discipline in LSPU?

Bot: Students shall be disciplined according to the rules and regulations of the University, with penalties ranging from suspension to expulsion.

TESTING AND EVALUATION

EVALUATION

As you can see, the model performed not as well as we expected it to be.

- **Accuracy:** Only 3.35% of the predictions were correct which the model is struggling to provide the correct output consistently.
- **Recall:** The model identifies only 3.35% of the true positives, meaning it's missing a significant portion of correct answers.
- **Precision:** Precision is perfect, meaning every prediction the model made as "correct" was right. Though this is the reason why you can't just rely on one performance metric.
- **F1 Score:** This combines precision and recall, and the low value is a clear imbalance between them. A perfect precision but very low recall drags the F1 score down.

```
You: metrics  
100% [██████] 179/179 [02:35<00:00, 1.15it/s]  
  
Model Metrics:  
Accuracy: 0.0335  
F1 Score: 0.0649  
Recall: 0.0335  
Precision: 1.0000
```

TESTING AND EVALUATION

EVALUATION

The poor performance can actually be seen during testing, as it would give out nonsensical and funny answers.

You: what is the policy for dropping a course in LSPU?
Bot: Dropping a course in LSPU is strictly prohibited.

You: what is the grading system used in LSPU?
Bot: The grading system is as follows: 1.00 (Good), 1.25 (Failure), 1.75 (Honorable), 2.25 (Recommended), 3.75 (Excellent), 4.75 (Outstanding), 5.75 (Very Satisfactory), 6.75 (Associate Dean),

You: what is the normal academic load for students in LSPU?
Bot: I'm sorry, I don't have information on that. Can you please rephrase or ask a different question?

You: what is the normal academic load for students?
Bot: A normal load is defined as the number of units or units required in a given academic year.

You: what is the policy regarding tattooing at LSPU?
Bot: Tattooing on exposed parts of the body is strictly prohibited, with penalties including suspension and disqualification.

You: what are the admission requirements for LSPU?
Bot: Admission requirements for LSPU include a minimum of five (5) years of study and a maximum of seven (8) years of service to the community.

On the bright side, it manages to give coherent answers and not gibberish.

CHALLENGES

CHALLENGES

- **Google Colab's GPU and Memory Limit**

- This was by far the biggest hurdle. Google Colab's has only around 4 hours worth of available memory and GPU training available for each user. Adjusting each parameter in the model training (making the epochs larger for more accuracy, increasing batch sizes, etc.) meant that we were consuming a lot of resources, and we were always getting cut off by Google.
- This resulted in the poor performance of the model.
- We could not possibly train the chatbot anymore than we did.



CHALLENGES

- **Dataset**

- The LSPU Student Handbook is 198 pages. Condensing this PDF was among our first challenges as it was dense. We found a way to convert it to a hierarchical format, but that wasn't appropriate for the T5 model, so we had to convert around 75% of the data into a question and answer format.



CONCLUSION

CONCLUSION

The LSPU Student AI-de was a success in some ways, but falls short in others. It's a fully functional instruction-based chatbot that can answer some questions about LSPU's policies in that it relies more in the patterns of the dataset and generates its own responses, but because of limitations such as Google Colab's finite amount of GPU and memory, it's still is in its early stages.

As for suggestions for potential improvements, the dataset could be cleaner, much more detailed, to make a good reference for the model. Also, using more resources (GPU and memory) would be a great help in improving the chatbot as it can be trained with adjustments in its parameters. A user interface would also improve the usage of the chatbot.





THANK YOU!