Natural language & Recommended system

DINH HOANG VIET PHUONG – 301123263

**LAB 02**

**Analysis report**

**1, 2. Phuong\_intents.js file**

Task: the “service order” process for burgers from a burger shop.

**3. Prepare and load the data**

a, Add to the intents file, either manually or through a program, 16 sixteen new intents related to the process assigned to you. Per intent add four expected patterns (utterance) based on what you think the user might ask per intent and three to four expected responses. Think carefully, about intents, patterns & responses.

{

"intents": [

{

"tag": "greeting",

"patterns": [ "Hi", "Hey", "Is anyone there?", "Hello", "How are you?" ],

"responses": [ "Hello", "Hi", "Hi there", "Hello how are you?" ]

},

{

"tag": "thanks",

"patterns": [ "Thanks", "Thank you", "That's helpful", "Thanks for the help" ],

"responses": [ "Happy to help!", "Any time!", "My pleasure", "You're most welcome!" ]

},

{

"tag": "menu\_inquiry",

"patterns": [

"What's on the menu?",

"Can I see the burger menu?",

"What burgers do you have?",

"Show me your burger selections."

],

"responses": [

"Here's our burger menu!",

"We have a variety of burgers, here's the menu.",

"Take a look at our delicious burger selections.",

"Here are all the burgers we offer."

]

},

{

"tag": "order\_placement",

"patterns": [

"I'd like to place an order",

"Can I order a burger?",

"I want to order a burger now",

"How do I order?"

],

"responses": [

"Sure, what would you like to order?",

"Of course! Which burger are you craving?",

"What can I get for you today?",

"Happy to take your order, what will it be?"

]

},

{

"tag": "customization\_options",

"patterns": [

"Can I customize my burger?",

"What customizations do you offer?",

"I want to add toppings, is that possible?",

"How can I customize my order?"

],

"responses": [

"Absolutely, you can customize your burger.",

"We offer various toppings and sauces for customization.",

"Sure, tell us how you'd like to customize your burger.",

"Yes, we have several customization options available."

]

},

{

"tag": "allergy\_inquiry",

"patterns": [

"Do you have any gluten-free options?",

"I have a peanut allergy, what can I eat?",

"Are there any allergen-free burgers?",

"What options do I have if I'm allergic to dairy?"

],

"responses": [

"We have gluten-free buns and allergen-free options.",

"Please inform us of your allergy, and we'll accommodate you.",

"We offer several allergen-free alternatives.",

"We can customize your order to be allergen-free."

]

},

{

"tag": "order\_status",

"patterns": [

"What's the status of my order?",

"Is my burger ready yet?",

"How long until my order is ready?",

"Can you check on my order?"

],

"responses": [

"Let me check on that for you.",

"Your order is being prepared, it won't be long.",

"It's almost ready, thank you for your patience.",

"I'll check the status, one moment please."

]

},

{

"tag": "delivery\_inquiry",

"patterns": [

"Do you offer delivery?",

"Can I get my burger delivered?",

"What are your delivery options?",

"How can I get my order delivered?"

],

"responses": [

"Yes, we offer delivery services.",

"We can deliver your burger right to your door.",

"Delivery options are available, just let us know your address.",

"Sure, we have several delivery methods."

]

},

{

"tag": "pickup\_inquiry",

"patterns": [

"Can I pick up my order?",

"Is there a pickup option?",

"How do I go about picking up my burger?",

"I prefer to pick up, is that possible?"

],

"responses": [

"Pickup is definitely an option, just let us know when.",

"Yes, you can pick up your order at our location.",

"We'll have your order ready for pickup.",

"Sure, just tell us the time you'll be here."

]

},

{

"tag": "payment\_options",

"patterns": [

"What payment methods do you accept?",

"Can I pay with a credit card?",

"Do you accept mobile payments?",

"How can I pay for my order?"

],

"responses": [

"We accept cash, credit cards, and mobile payments.",

"Credit cards and mobile payments are welcome here.",

"You can pay using any major credit card or mobile payment service.",

"Multiple payment methods are accepted for your convenience."

]

},

{

"tag": "order\_modification",

"patterns": [

"Can I change my order?",

"I need to add something to my order.",

"I want to remove an item from my order.",

"Is it too late to modify my order?"

],

"responses": [

"Sure, what changes do you need?",

"Not a problem, what would you like to add or remove?",

"We can modify your order, what do you need?",

"It's not too late, what adjustment is required?"

]

},

{

"tag": "order\_cancellation",

"patterns": [

"I need to cancel my order",

"Can I cancel my burger order?",

"How do I cancel my order?",

"Is it possible to cancel my order now?"

],

"responses": [

"I've cancelled your order, no worries.",

"Your order has been cancelled as requested.",

"Sure, I've taken care of canceling your order.",

"Cancellation confirmed, anything else I can help with?"

]

},

{

"tag": "feedback\_request",

"patterns": [

"How can I give feedback?",

"I'd like to leave a review",

"Where can I comment about my experience?",

"I want to provide feedback on my order"

],

"responses": [

"We'd love to hear your feedback, here's how.",

"Thank you for wanting to leave a review, you can do so here.",

"Your feedback is valuable, please share it with us here.",

"Here's the link to leave your comments and feedback."

]

},

{

"tag": "loyalty\_program",

"patterns": [

"Do you have a loyalty program?",

"How can I join the loyalty program?",

"What are the benefits of your loyalty program?",

"Tell me about your loyalty program"

],

"responses": [

"Yes, we have a loyalty program for our regular customers.",

"Joining our loyalty program is easy, here's how.",

"Our loyalty program offers discounts and freebies.",

"Here are all the perks of our loyalty program."

]

},

{

"tag": "nutritional\_info",

"patterns": [

"Can I get nutritional information for your burgers?",

"What are the calorie counts of your burgers?",

"Do you provide information on allergens?",

"I need the nutritional details of your menu"

],

"responses": [

"Sure, here's the nutritional information.",

"Each burger's calorie count is listed here.",

"We provide detailed allergen and nutritional info.",

"You can find nutritional details on our menu."

]

},

{

"tag": "special\_offers",

"patterns": [

"Any special offers today?",

"What deals do you have right now?",

"Tell me about your current promotions",

"Do you have any discounts?"

],

"responses": [

"Check out our current special offers!",

"We have some great deals available right now.",

"Here are today's special promotions.",

"Yes, we're offering discounts on these items today."

]

}

]

}

b, Record the details of the intents, utterances, and responses in your analysis report with a clarification why you chose each. (Table format)

|  |  |  |  |
| --- | --- | --- | --- |
| **INTENT** | **EXAMPLE UTTERANCES** | **EXAMPLE RESPONSES** | **RATIONALE** |
| **Greeting** | "Hi", "Hello" | "Hello, how are you?" | Chosen for its broad applicability and friendly tone, initiating a positive customer interaction. |
| **Thanks** | "Thank you" | "Any time!" | Acknowledges and reciprocates the appreciation, fostering goodwill. |
| **Menu Inquiry** | "What burgers do you have?" | "Here's our burger menu!" | Directly addresses the customer's request for information, encouraging further interaction. |
| **Order Placement** | "Can I order a burger?" | "Of course! Which burger are you craving?" | Invites the customer to specify their choice, facilitating a transaction. |
| **Customization Options** | "Can I customize my burger?" | "Absolutely, you can customize your burger." | Confirms the availability of options, promoting customer satisfaction through personalization. |
| **Allergy Inquiry** | "Do you have any gluten-free options?" | "We have gluten-free buns and allergen-free options." | Provides specific information to reassure customers about dietary concerns. |
| **Order Status** | "Is my burger ready yet?" | "It's almost ready, thank you for your patience." | Communicates progress and appreciates the customer's patience, enhancing service perception. |
| **Delivery Inquiry** | "Do you offer delivery?" | "Yes, we offer delivery services." | Clearly confirms service availability, facilitating convenience for the customer. |
| **Pickup Inquiry** | "Can I pick up my order?" | "Pickup is definitely an option, just let us know when." | Affirms the service and prompts for further action, streamlining the process. |
| **Payment Options** | "What payment methods do you accept?" | "We accept cash, credit cards, and mobile payments." | Informs customers of their flexibility, easing the payment process. |
| **Order Modification** | "Can I change my order?" | "Sure, what changes do you need?" | Opens a dialogue for modification, demonstrating flexibility and customer service. |
| **Order Cancellation** | "I need to cancel my order" | "Your order has been cancelled as requested." | Responds to the customer's request efficiently, ensuring clarity and closure. |
| **Feedback Request** | "How can I give feedback? | "We'd love to hear your feedback, here's how." | Encourages customer engagement and shows value for their input. |
| **Loyalty Program** | "Do you have a loyalty program?" | "Yes, we have a loyalty program for our regular customers." | Highlights benefits for repeat customers, encouraging customer retention. |
| **Nutritional Info** | "Can I get nutritional information for your burgers?" | "Sure, here's the nutritional information." | Addresses health-conscious consumers' needs, promoting transparency. |
| **Special Offers** | "Any special offers today?" | "Check out our current special offers!" | Generates interest in promotions, potentially increasing sales. |

c, Write code to read your JSON file, and store the elements into a group of lists. (Hint: you will need four lists).

A screen shot of a computer

Description automatically generated

A computer screen with text and images

Description automatically generated

* Data Loading: The script defines a file path to a JSON file (phuong\_intents.json) which contains structured data about different intents.
* Data Extraction: The script then initializes empty lists to hold tags (intents), patterns (questions), responses, and contexts. It opens the specified JSON file, loads its content, and iterates through each intent. For each intent, it appends its patterns and tag to the patterns and tags lists, respectively. Responses are added to the responses list (though the method used seems to be intending to append each response individually but instead extends the list by each character in the response due to a missing loop or incorrect method usage). Contexts are also collected, with a default value of None if not specified.

**4. Pre-processing**

A computer screen shot of a program code

Description automatically generated

**Label Encoding:**

* Utilizes LabelEncoder from sklearn.preprocessing to convert the list of tags (representing intents) into numerical labels. This is essential for machine learning models that require numerical inputs.
* intent\_labels are obtained by fitting the label encoder to the tags and transforming them into a numerical format.
* Determines the number of unique classes (num\_classes) by finding the maximum value in intent\_labels and adding 1, ensuring a count that starts from 0.

**Tokenization:**

* Initializes a Tokenizer from Keras' preprocessing tools, specifying a maximum number of words (num\_words=800) and an out-of-vocabulary token (oov\_token="<OOV>"). This prepares the tokenizer to only consider the top 800 words based on frequency and to handle unseen words by replacing them with <OOV>.
* Fits the tokenizer on the patterns (questions) to create a word index (word\_index), which is a dictionary mapping words to their numerical representation.
* Transforms the text patterns into sequences of tokens (sequences) using the trained tokenizer, effectively converting each pattern into a sequence of numbers corresponding to the words in the pattern.
* Pads these sequences using pad\_sequences to ensure that all input sequences have the same length (maxlen=45), with padding applied at the end (padding='post'). This uniform length is crucial for training neural network models, which require fixed-size inputs.

**5. Deep learning training**

a,

A screenshot of a computer program

Description automatically generated

**Data Splitting:**

* The pre-processed padded sequences and their corresponding numerical labels are split into training and validation sets using train\_test\_split from sklearn.model\_selection.
* The split is done with 20% of the data allocated for validation (test\_size=0.2) and a random\_state of 63 (the last 2 digits of my student number) to ensure reproducibility.

**Model Definition:**

* A Sequential model is defined using Keras, indicating a linear stack of layers.
* The model architecture includes:
  + An Embedding layer as the first layer, configured with 800 input dimensions (vocabulary size), an output dimension of 25, and an input length of 45. This layer is crucial for processing text by converting token sequences into dense vectors of fixed size.
  + A GlobalAveragePooling1D layer to reduce the dimensionality of the feature maps and prepare for the dense layer.
  + Three Dense layers with different configurations to learn the mappings from inputs to outputs. The first two have 16 and 10 units with 'relu' and 'sigmoid' activations, respectively, and the third has 8 units with 'relu' activation, aiming to progressively refine the learned representations.
  + A final Dense layer with a number of units equal to num\_classes and 'softmax' activation to output probabilities for each class.

**Model Compilation:**

* The model is compiled with the sparse\_categorical\_crossentropy loss function, suitable for multi-class classification tasks where each class is mutually exclusive.
* The adam optimizer is chosen for training, a popular choice for deep learning models due to its adaptive learning rate capabilities.
* The metric 'accuracy' is used to monitor the training and validation process.

b,

A screenshot of a computer

Description automatically generated

**Model Training:**

* The model is trained using the training data with validation on the validation set. Training is set for 500 epochs, a relatively high number, indicating that the model will go through the training data 500 times, adjusting weights to minimize loss.
* validation\_data is provided to evaluate the model’s performance on unseen data after each epoch, providing insights into how well the model generalizes.

Accuracy Reporting:

* After training, the accuracy of the model on the training data is extracted from the training history and converted into a percentage.
* The accuracy is printed out, providing a quantitative measure of how well the model has learned to classify the intents based on the patterns.

c,

A screenshot of a computer program

Description automatically generated

**Improvement in Accuracy with More Training:** The model's accuracy improves from 88.46% at 500 epochs to 100.00% at 1000 epochs, showcasing the benefit of additional training. This significant increase suggests that the model has effectively learned and generalized from the provided data, capturing the underlying patterns accurately.

A screenshot of a computer program

Description automatically generated

**Potential Overfitting Concerns:** Achieving 100% accuracy raises concerns about potential overfitting, where the model may have learned the training data too well, including its noise and outliers. This could negatively impact the model's ability to generalize to new, unseen data, especially if the validation set lacks sufficient challenge or diversity.

**The Importance of Epoch Number in Training:** The stark difference in accuracy underscores the critical role of epoch count in training neural networks. Insufficient epochs may result in underfitting, whereas too many epochs risk overfitting. Striking the right balance is essential for optimal model performance.

**Decision Against Early Stopping:** For this specific comparison between training for 500 and 1000 epochs, early stopping was intentionally not utilized. The purpose here is to observe the full impact of increasing epochs on model accuracy without prematurely halting the training process. This approach allows for a clearer comparison but also highlights the importance of being mindful of potential overfitting.

**6. Testing the bot**

a, b, c, d,

A screen shot of a computer program

Description automatically generated

**Save Tokenizer:**

* Opens a file (tokenizer.pickle) in write-binary (wb) mode at the specified path (D:\\Download\\).
* Uses pickle.dump to serialize the tokenizer object into the file. The protocol=pickle.HIGHEST\_PROTOCOL argument specifies using the latest version of the pickle protocol, offering better performance and smaller file sizes.

**Save Encoder:**

* Similarly, opens a file (label\_encoder.pickle) in write-binary mode at the same path.
* Serializes the label encoder object into this file using pickle.dump, with the same protocol argument for efficiency.

**Save Model:**

* The trained model is saved directly to a directory (D:\\Download\\model) using the model.save method. This method not only saves the model's architecture and weights but also the training configuration (loss, optimizer) and the state of the optimizer, allowing for resuming training from where it was left off.

A screen shot of a computer program

Description automatically generated

This code snippet is a comprehensive demonstration of how to load pre-trained components (tokenizer, label encoder, and a neural network model) and utilize them within a simple chatbot framework to predict user intent based on their input. The snippet ends with a practical application: an interactive loop where the user can chat with the bot, and the bot responds based on the predicted intent of the user's message. Here's a breakdown:

**Loading Pre-Trained Components:**

* The tokenizer and label encoder are loaded from .pickle files using pickle.load. This restores their state from when they were saved, including the tokenizer's word index and the label encoder's class labels.
* The trained model is loaded from a directory using tf.keras.models.load\_model. This restores the model's architecture, weights, optimizer state, and training configuration.

**Loading Intents JSON File:**

* Loads the intents JSON file into the intents variable. This file contains the intents, patterns, and responses that the chatbot can recognize and use to respond to user inputs.

**Predicting Intent:**

* The predict\_intent function preprocesses the input text (tokenization and padding) and then uses the loaded model to predict the intent of the text. The prediction is a probability distribution across all possible intents, and the function selects the intent with the highest probability.
* It then uses the label encoder to find the corresponding intent label for the predicted intent index.

**Chatbot Interaction Loop:**

* Initiates an interactive chat loop where the user inputs text, and the chatbot responds. The user can exit the loop by typing "bye".
* For each user input, the predict\_intent function is called to determine the intent of the message. The chatbot then searches the loaded intents for a matching tag and randomly selects one of the corresponding responses to print.

**Key Points and Considerations:**

* Flexibility and Extendibility: This setup allows for easy updates and modifications to the chatbot's responses and intents by simply updating the intents JSON file and retraining the model if necessary.
* Random Response Selection: The chatbot selects a random response for the predicted intent, which can help make the chatbot's interactions feel more varied and less robotic.
* Error Handling: While not explicitly shown, implementing error handling (for unrecognized inputs, loading errors, etc.) would be essential for a robust chatbot application.
* Customization Potential: The framework can be expanded with more sophisticated features, such as context-aware responses, handling of entities (like names, dates, etc.), and integration with external APIs for more dynamic interactions.
* This code forms a basic yet functional chatbot capable of understanding predefined intents and responding accordingly, serving as a foundation for more complex conversational AI applications.

e, Ways to Improve the Bot

* Contextual Understanding: Enhance the bot's ability to maintain and understand the context of a conversation across multiple exchanges. Implementing a context management system allows the bot to deliver more relevant responses based on the conversation history.
* Entity Recognition: Integrate Named Entity Recognition (NER) to identify and understand specific entities within user inputs, such as dates, locations, names, etc. This allows for more personalized and accurate responses.
* Sentiment Analysis: Incorporate sentiment analysis to gauge the user's mood or tone from their messages. The bot can use this information to tailor its responses to be more empathetic or aligned with the user's emotional state.
* Multi-Language Support: Expand the bot's capabilities to understand and respond in multiple languages, broadening its accessibility and user base.
* Personalization: Implement user profiling to tailor conversations and recommendations based on individual user preferences, history, and behavior.
* Continuous Learning: Establish a feedback loop where the bot learns from its interactions. This can be through direct user feedback, monitoring user engagement with responses, or supervised retraining with updated datasets.
* Integration with External APIs and Services: Enable the bot to fetch information in real-time from external sources (e.g., weather forecasts, news updates, booking services) to provide more dynamic and useful responses.
* Voice Recognition and Synthesis: Incorporate speech-to-text and text-to-speech capabilities to allow for voice interactions, making the bot more accessible and enhancing user experience.

f, External Dataset Suggestions

* Reddit Conversations Dataset: Reddit, being a vast platform with discussions on countless topics, offers datasets that can be used to train the bot in understanding a wide range of intents and formulating responses. This can be particularly useful for informal conversation styles.
* Customer Support Datasets from Companies: Many companies release datasets of customer support interactions. These datasets can help train the bot in handling specific queries, complaints, and requests, making it suitable for customer service applications.
* The Stanford Natural Language Inference (SNLI) Corpus: The SNLI corpus is a collection of sentence pairs labeled for entailment, contradiction, and neutrality. This dataset can help improve the bot's understanding of language nuances and relationships between sentences, enhancing its ability to process user inputs more accurately.
* Multi-Domain Wizard-of-Oz dataset (MultiWOZ): A fully labeled collection of human-human written conversations spanning multiple domains and scenarios. Training on MultiWOZ can significantly enhance the bot's ability to handle complex, multi-turn conversations across various subjects.
* Google's Dataset Search: A tool that enables the search of datasets stored across the web. Using this tool, you can find specific datasets tailored to your needs, such as domain-specific conversations, cultural nuances, or languages, to augment your existing dataset for a more comprehensive training process.