**Input.**

*Read the contents of the intents.json file containing predefined intents for the chatbot.*

*Parse the JSON data to extract intents, patterns, and responses.*

**begin**

1. Tokenization - Split the text into individual words or tokens. Mathematical Representation: Let the input text data be represented as a set of sentences, *S = {s1, s2, s3…, sn}*. Tokenize each sentence si to obtain individual words/tokens: *si 🡪 {wi1, wi2, wim}*, where m is the number of words in sentence si.
2. Lowercase + Stemming - Convert all words to lowercase to ensure uniformity. Perform stemming to reduce words to their root/base form. Lowercasing: *wij = lower (wij).* Stemming: *wij = stem(wij).*
3. Stemmed words - Remove punctuation marks from the stemmed words. wij = remove\_ punctuation(wij).
4. Bag-of-Words Representation – Create a vocabulary containing unique words from the processed tokens. Create a vocabulary: *V 🡪 {v1, v2, …, vk}* where *k* is the number of unique words. Construct a mathematical representation of each sentence using a vector of word counts. Represent each sentence si as a vector vi of words counts: *vi = [c(v1 , si), c(v2 , si), …, c(vk , si)]*, where *c(vj , si)* represents the count of word vj in sentence si.
5. Training with Feedforward Neural Network- Create a feedforward neural network model. Input the Bag-of-Words vectors into the network for training. Train the neural network using backpropagation and optimize the weights using a suitable loss function Training the network involves updating weights **W** and biases **b** through iterations using gradient descent: *Wnew = Wold – α. ∇L(Wold)*, where **α** is the learning rate and **L** is the loss function.
6. Create Training Data - Generate bag-of-words representations for patterns. Create input-output pairs for training.
7. Prepare for Neural Network - Shuffle and convert training data to a NumPy array. Separate features and labels (train\_x, train\_y).
8. Define Neural Network Model - Create a Sequential model with layers: Dense(128, 'relu'), Dropout(0.5), Dense(64, 'relu'), Dropout(0.5). Output layer with softmax activation.
9. Compile and Train Model - Compile the model using SGD optimizer (lr=0.01, decay=1e-6, momentum=0.9). Use categorical crossentropy loss function.
10. Save the model.

**End**