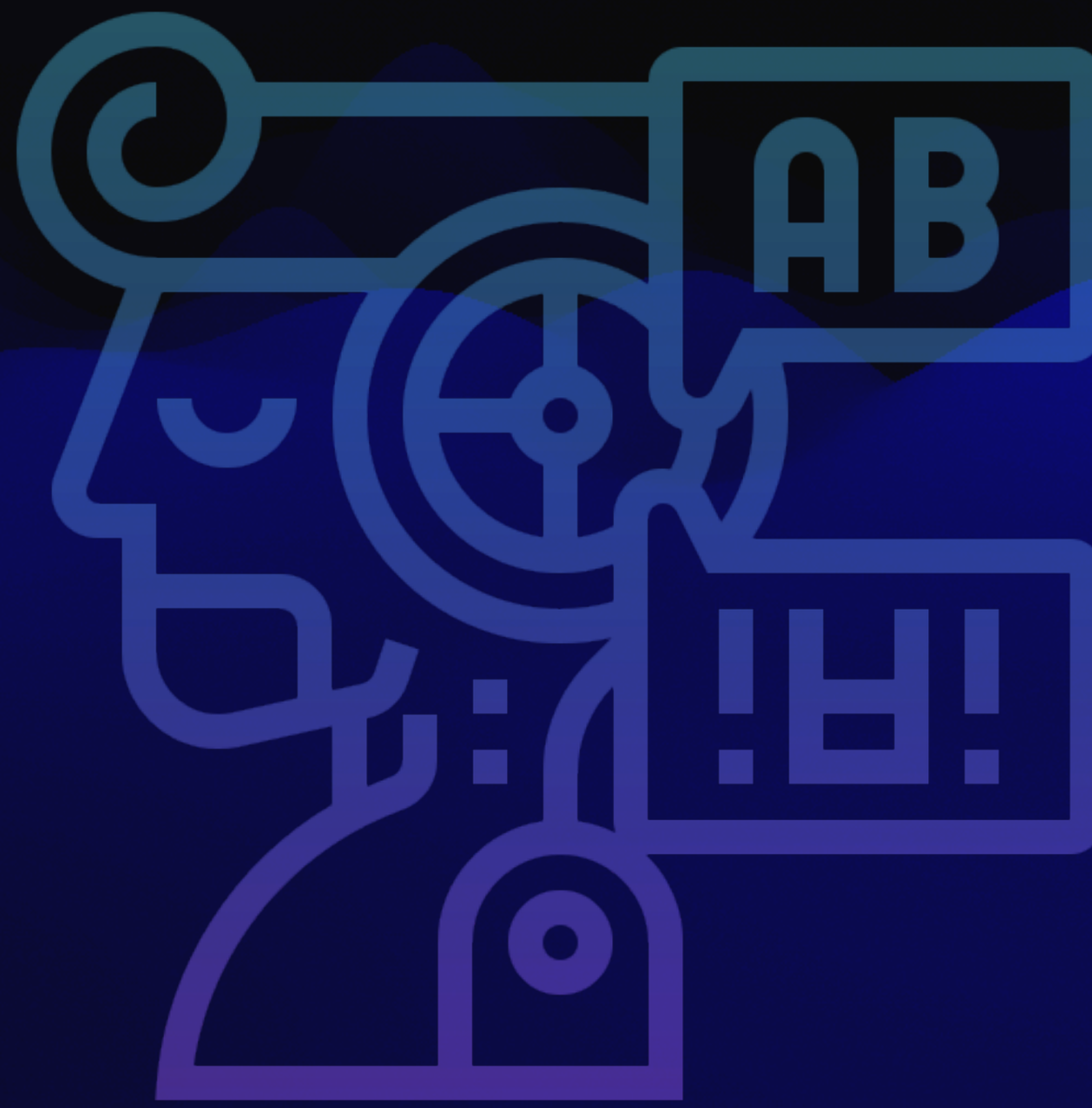
Chatbot Crafter



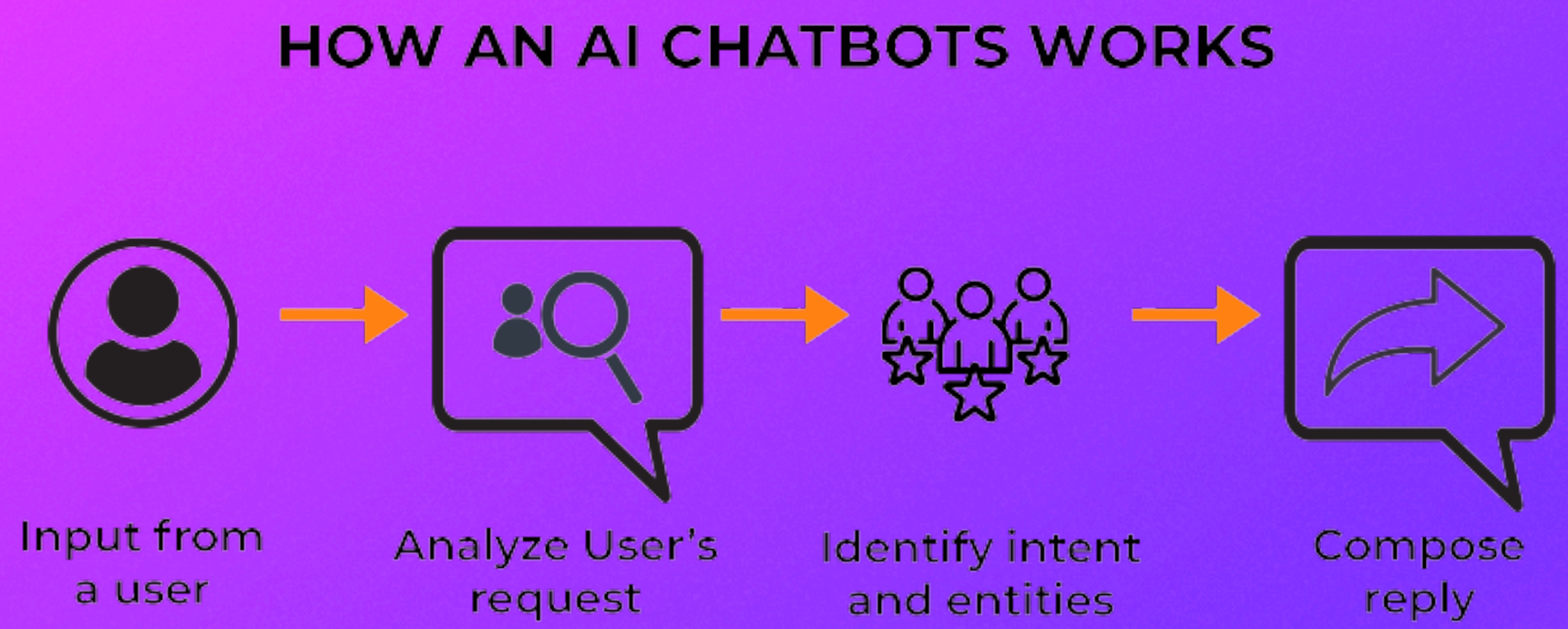
What We Learn?

- What is ChatBot?
- What is NLP?
- NLP tools in Python
- Introduction to Deep Learning
- Simple Chatbot using NLTK and Tensorflow(Python)
- RASA NLU - Rock-Paper-Scissors
- OpenAI- ChatGPT Personalised Chatbot

WHAT IS CHATBOT?

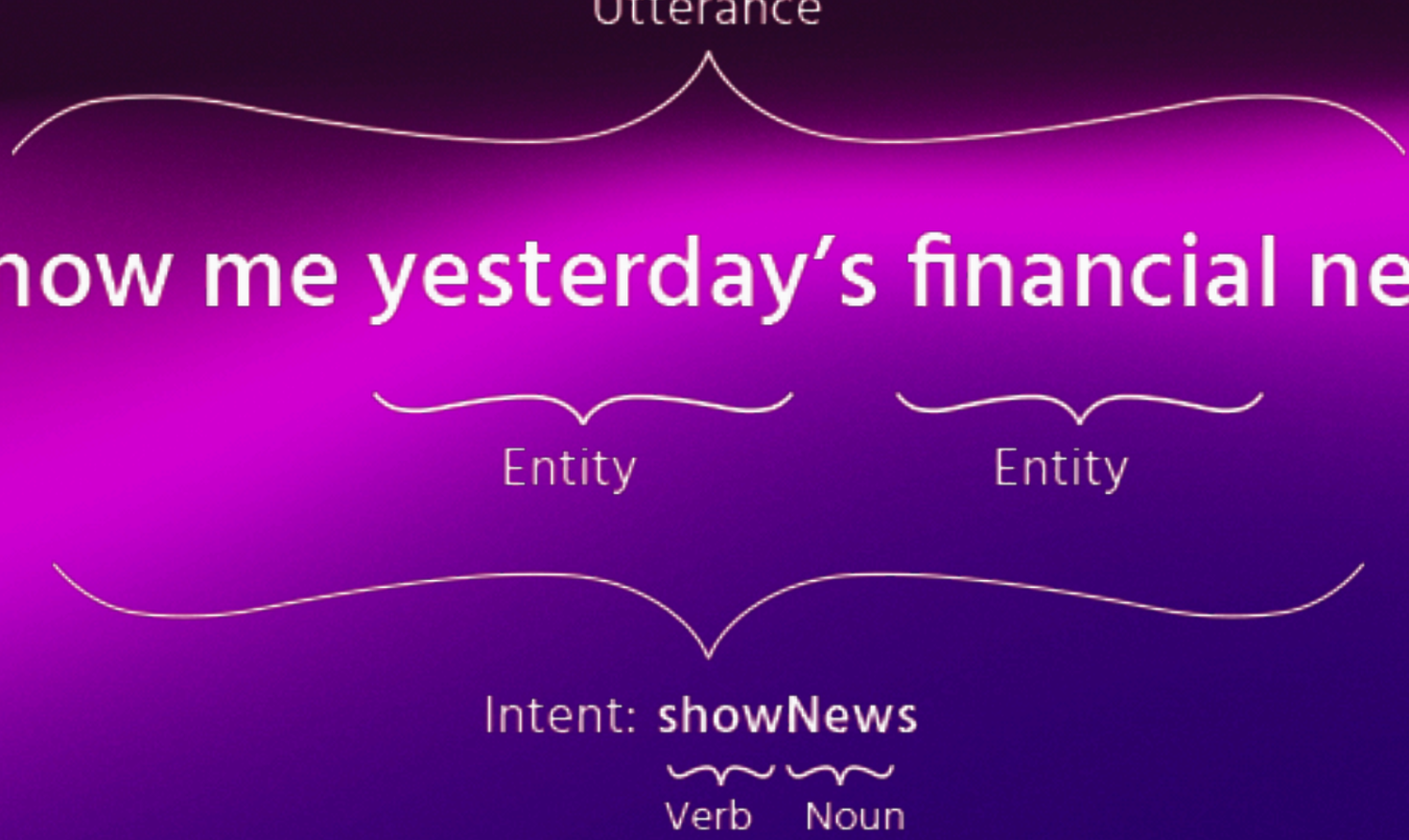
They are simulations that can:

- Understand human languages
- Process requests
- Interact back with humans



INTENT and ENITY

- Intent - Intention, or purpose of the user in the conversational flow.
- Entity - A data point or value which you can extract from a conversation



Utterances

Extracted Intents and Entities

S

Stephan E Strange

Could you book a restaurant in Manhattan?

Apr 2, 2021

Intent:
RestaurantBooking

Entities:
Location: Manhattan

W

Wall E West

Could you book a seafood restaurant near me?

Apr 2, 2021

Intent:
RestaurantBooking

Entities:
Location: near me
Cuisine: seafood

P

Peter Parkeur

Could you make a reservation at Red Lobster tonight?

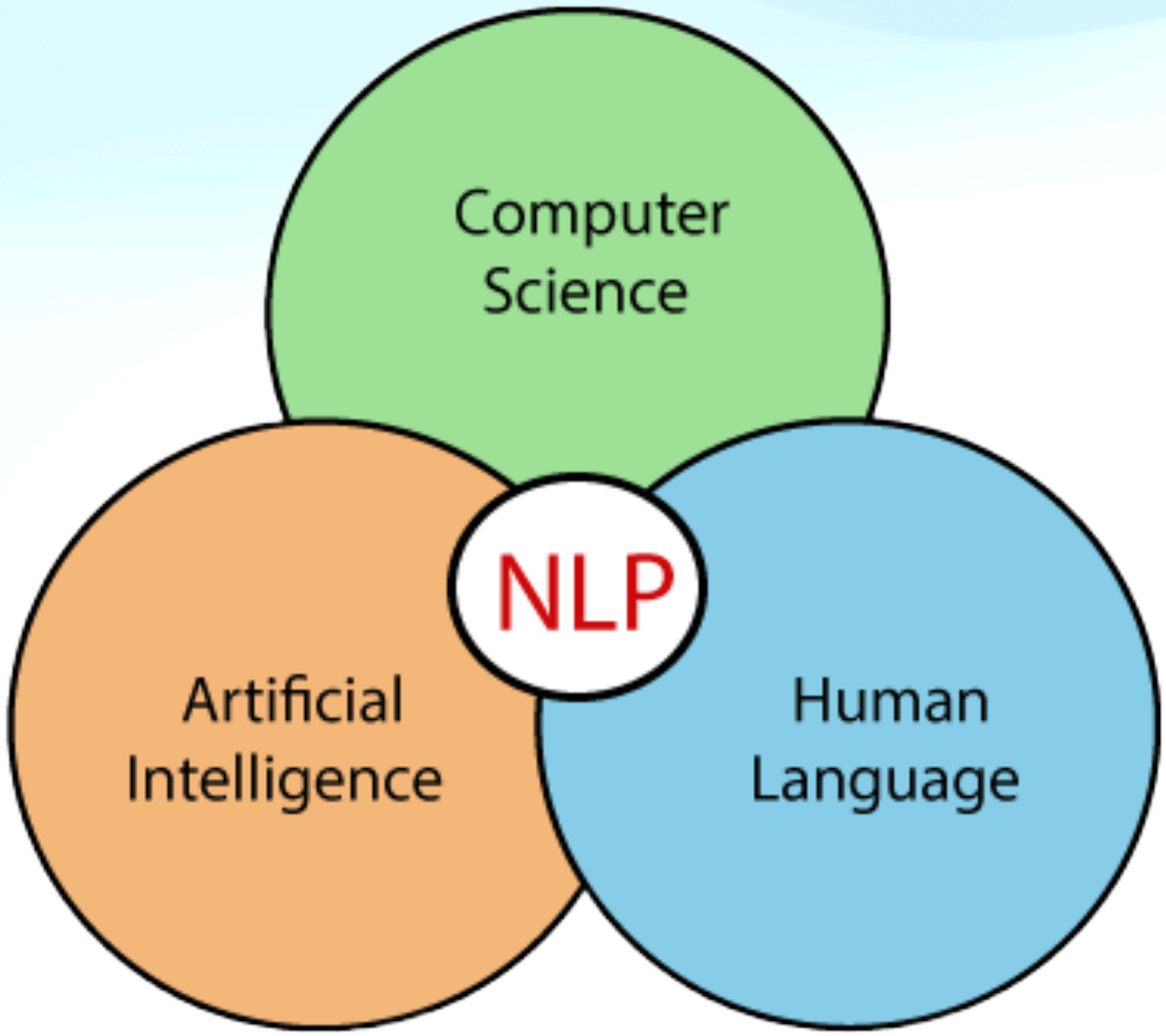
Apr 2, 2021

Intent:
RestaurantBooking

Entities:
Place name: Red Lobster
Time: tonight

Natural Language Processing (NLP)

It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages.



Applications:

- Translation
- Automatic summarization
- Named Entity Recognition (NER)
- Speech recognition
- Relationship extraction
- Topic segmentation

Components of NLP

1. Natural Language Understanding (NLU): helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

    - It is used to map the given input into useful representation.
    - It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG) : It acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

    - generating the responses of chatbots and voice assistants such as Google's Alexa and Apple's Siri;
    - automating lead nurturing email, messaging and chat responses;
    - aggregating and summarizing news reports;

Google Colab

colab.research.google.com

Working with Text in Python

- len
- split
- lower
- capitalize
- upper
- count
- startswith
- endswith

NLTK- Natural Language Toolkit

- Text Preprocessing Level 1 :
    - Tokenization
    - StopWords
    - Stemming
    - Lemmatization
    - N-gram
    - Disambiguating word meanings
- Text Preprocessing Level 2:
    - Bag Of Words
    - TFIDF
    - Word2vec

https://github.com/alexjolly28

nltk.download(‘all’)

Corpus: (Plural: Corpora) a collection of written texts that serve as our datasets.

Tokenization

- Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens.
- tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization.
- This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.
- word tokenize Word tokenization is the process of splitting up “sentences” into “words”
- sentence tokenize. Sentence tokenization is the process of splitting up strings into “sentences”

Removing Stop Words

Stopwords are non-content words that primarily has only grammatical function

Often, we want to remove the punctuations from the documents too.

```
from nltk.corpus import stopwords
from string import punctuation
```

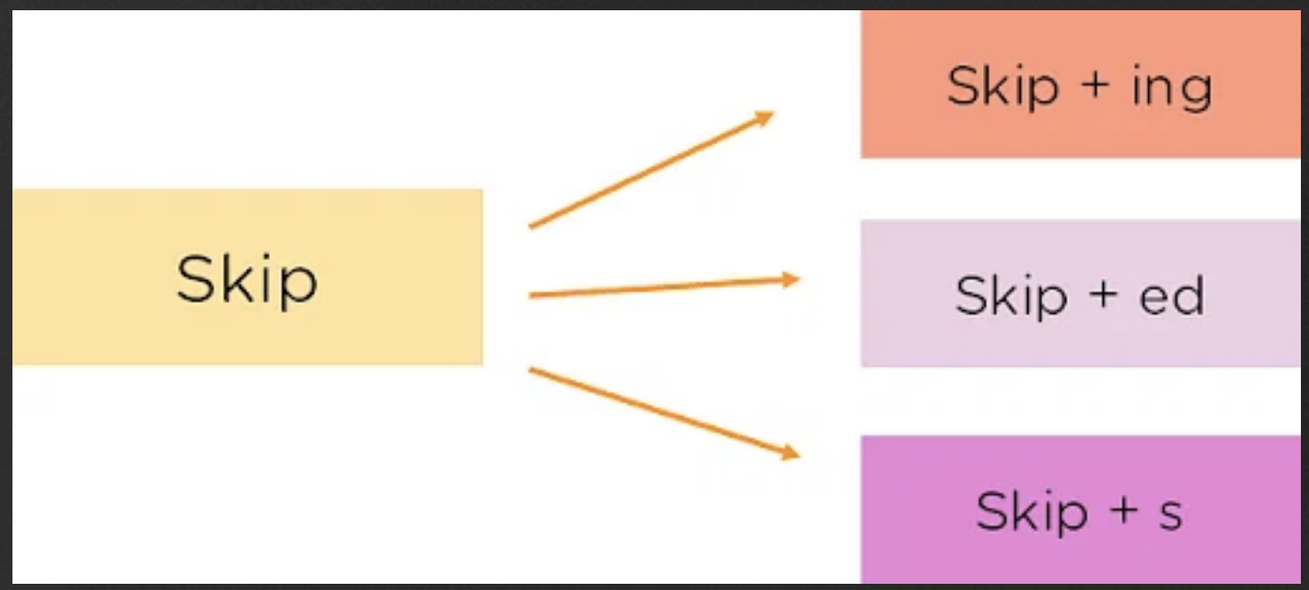
Normalizing Text

Stemming and Lemmatization: both are used to generate root form of derived (inflected) words

Stemming

It is the process of reducing the words to their word stem or root form. The objective of stemming is to reduce related words to the same stem even if the stem is not a dictionary word.

Stemming is the process of removing the last few characters of a given word, to obtain a shorter form, even if that form doesn't have any meaning.



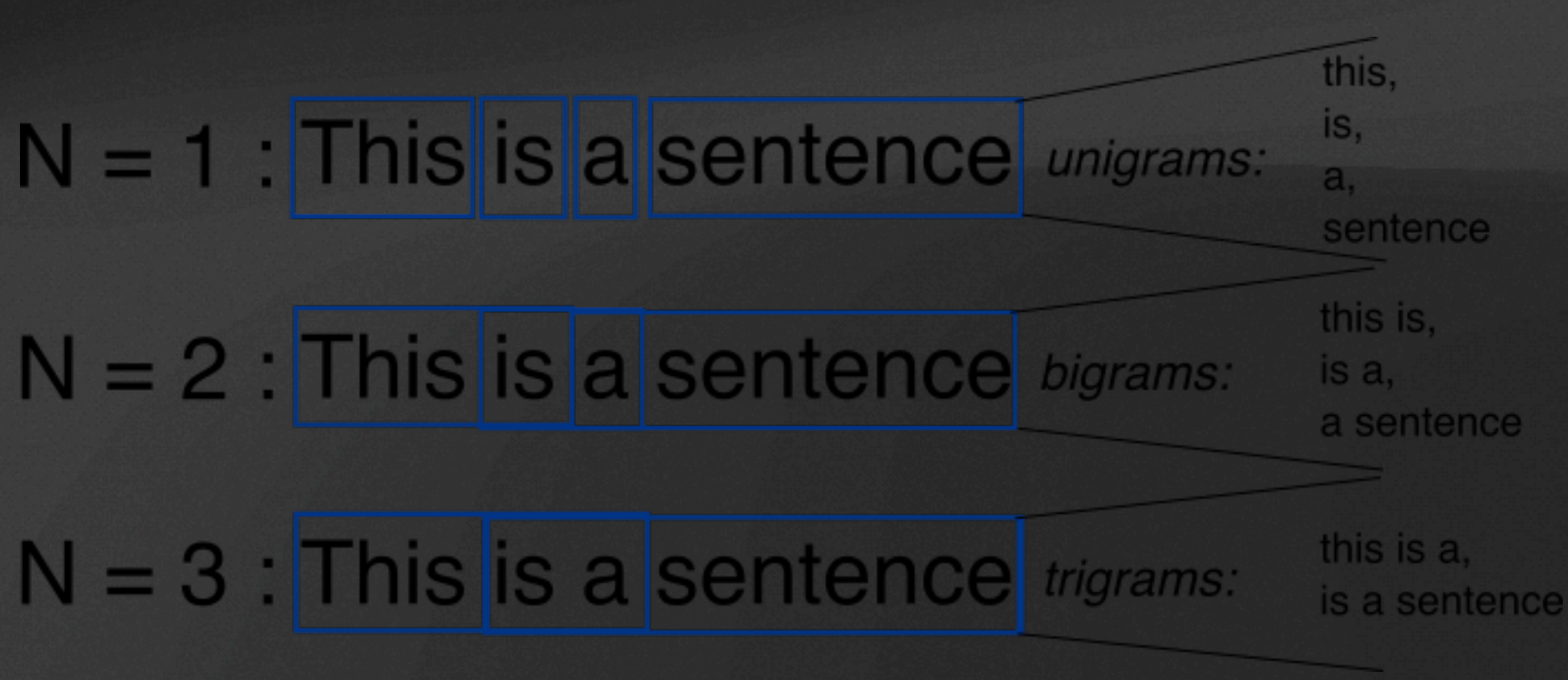
Lemmatization

Unlike stemming, lemmatization reduces words to their base word, reducing the inflected words properly and ensuring that the root word belongs to the language

- Stemming: Trying to shorten a word with simple regex rules , Stemmers are much simpler, smaller, and usually faster than lemmatizers
- Lemmatization: Trying to find the root word with linguistics rules (with the use of regexes)
- Stemming has its application in Sentiment Analysis while Lemmatization has its application in Chatbots, human-answering.

N gram

N-grams are continuous sequences of words or symbols or tokens in a document.
In technical terms, they can be defined as the neighbouring sequences of items in a document.



Disambiguating word meanings

words have different meanings based on the context of its usage in the sentence.

If we talk about human languages, then they are ambiguous too because many words can be interpreted in multiple ways depending upon the context of their occurrence.

Word sense disambiguation, in natural language processing (NLP), may be defined as the ability to determine which meaning of word is activated by the use of word in a particular context.

Vectorization

- Text vectorization is the process of converting text into a numerical representation that can be processed by machine learning algorithms.

- These vectors can then be used as input to a machine learning model for tasks such as classification, clustering, or regression.

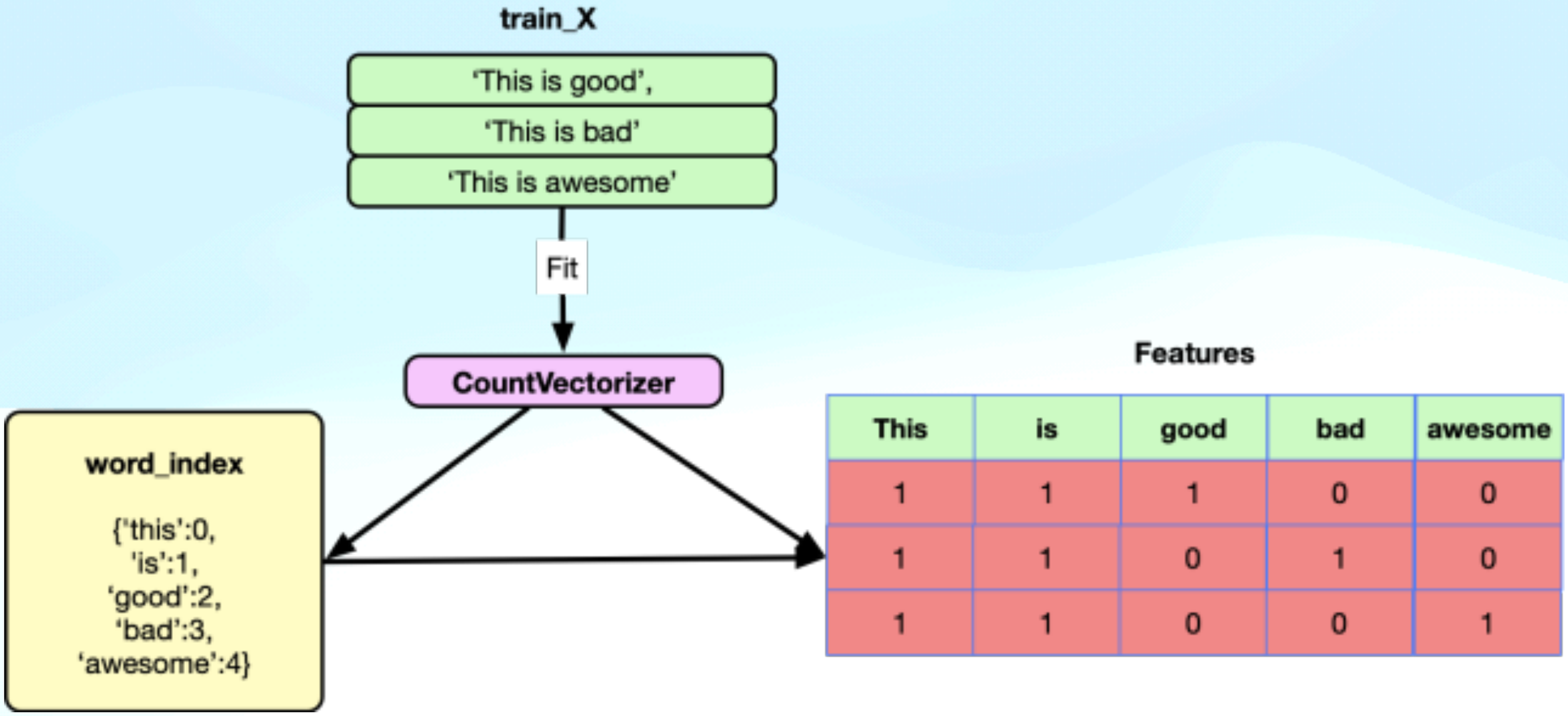
Bag of Words

Bag of Words takes a document from a corpus and converts it into a numeric vector by mapping each document word to a feature vector for the machine learning model.

a piece of text and count the frequency of the words in that text.

Imagine a corpus with thousands of articles. In such a case, the number of unique words in a dictionary can be thousands. If one document contains 10% of the unique words, the corresponding embedding vector will still contain 90% zeros.

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0


train_X
'This is good',
'This is bad'
'This is awesome'
Fit
CountVectorizer
word_index
{'this':0,
'is':1,
'good':2,
'bad':3,
'awesome':4}
Features
This
is
good
bad
awesome
1
1
1
0
0
1
1
0
1
0
1
1
0
0
1

TF-IDF Term Frequency - Inverse Document Frequency

Term frequency-inverse document frequency (TF-IDF) gives a measure that takes the importance of a word into consideration depending on how frequently it occurs in a document and a corpus.

TF-IDF is a product of two values: Term Frequency (TF) and Inverse Document Frequency (IDF).

Bag of Words: Converting words to numbers with no semantic information.

TF-IDF: It is also converting the words to numbers or vectors with some weighted information.

Term Frequency

Term frequency denotes the frequency of a word in a document.

TF of a term or word is the number of times the term appears in a document compared to the total number of words in the document.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

Inverse Document Frequency

It measures the importance of the word in the corpus. It measures how common a particular word is across all the documents in the corpus.

$$IDF(\textbf{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \textbf{term}\ in\ it}\right)$$

wordToVec

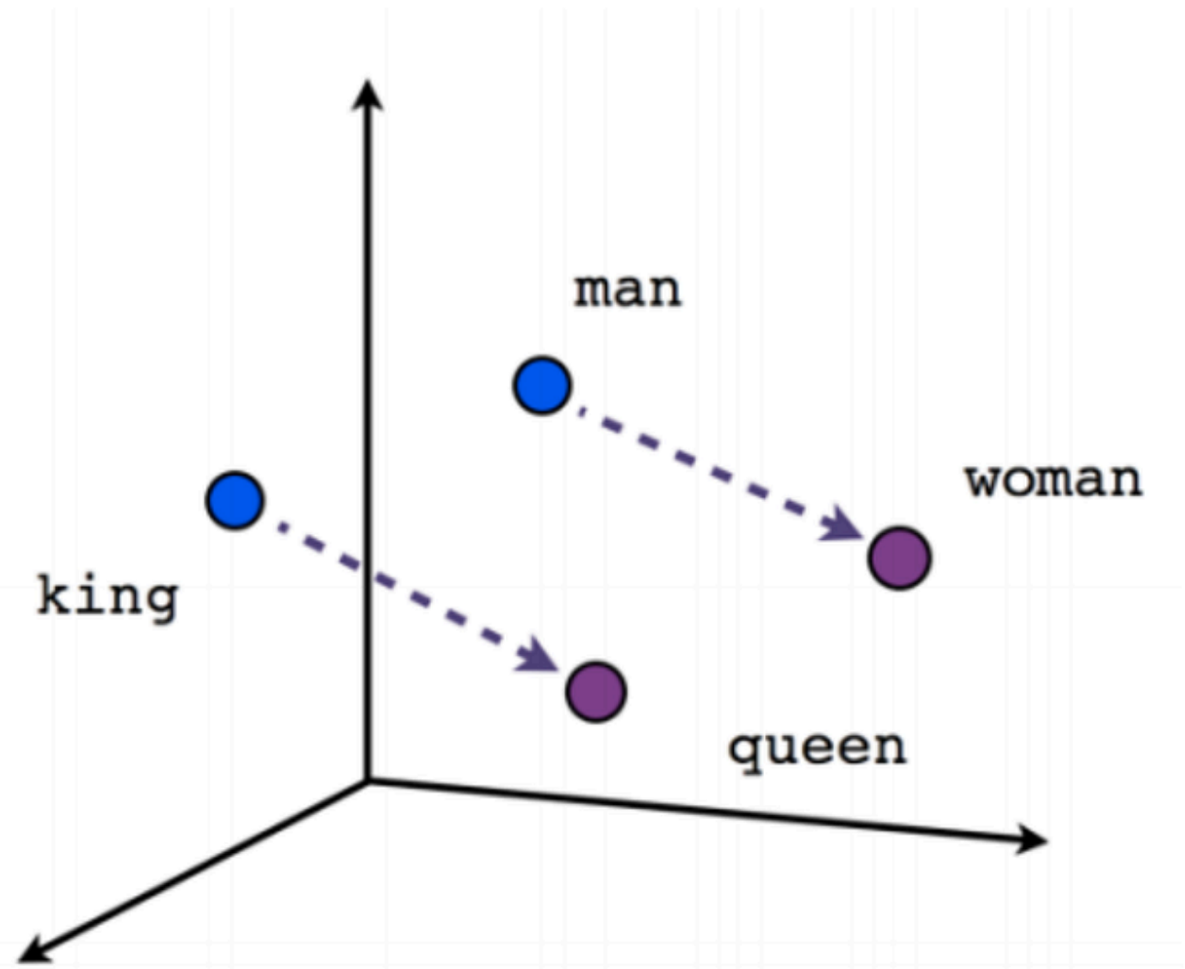
The effectiveness of Word2Vec comes from its ability to group together vectors of similar words

Word2Vec approach uses deep learning and neural networks-based techniques to convert words into corresponding vectors in such a way that the semantically similar vectors are close to each other in N-dimensional space, where N refers to the dimensions of the vector.

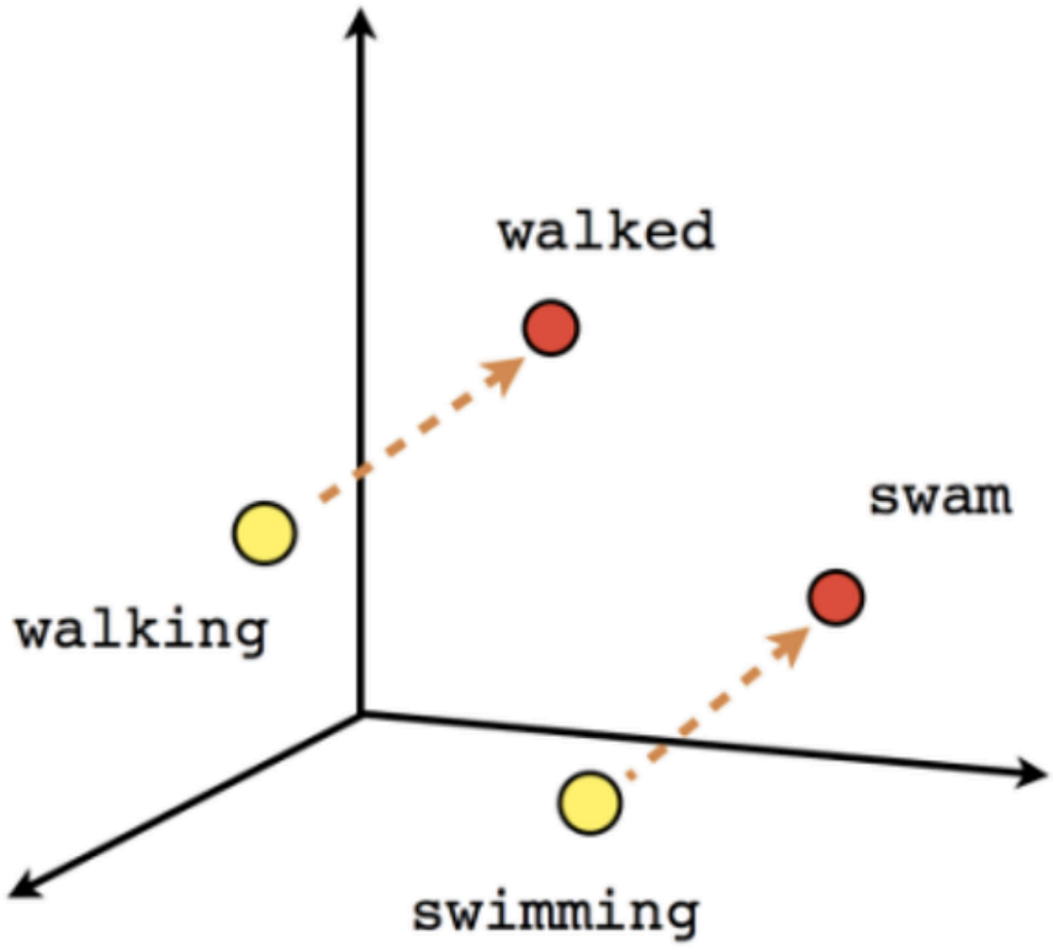
Word2Vec retains the semantic meaning of different words in a document

The context information is not lost.

size of the embedding vector is very small.

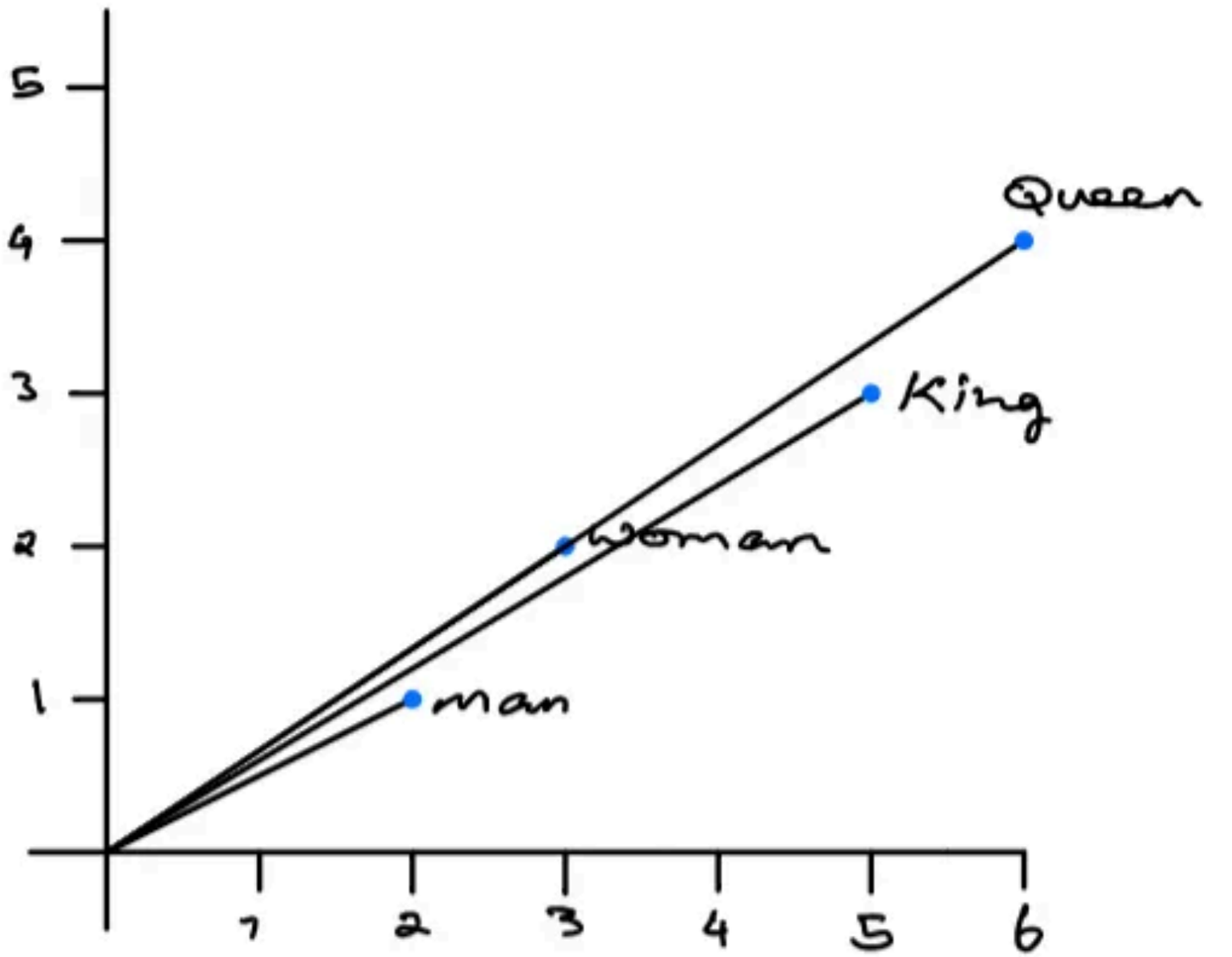


Male-Female



Verb tense

```
King    -    Man    +    Woman    =    Queen
[5,3]   -    [2,1]  +    [3, 2]   =    [6,4]
```

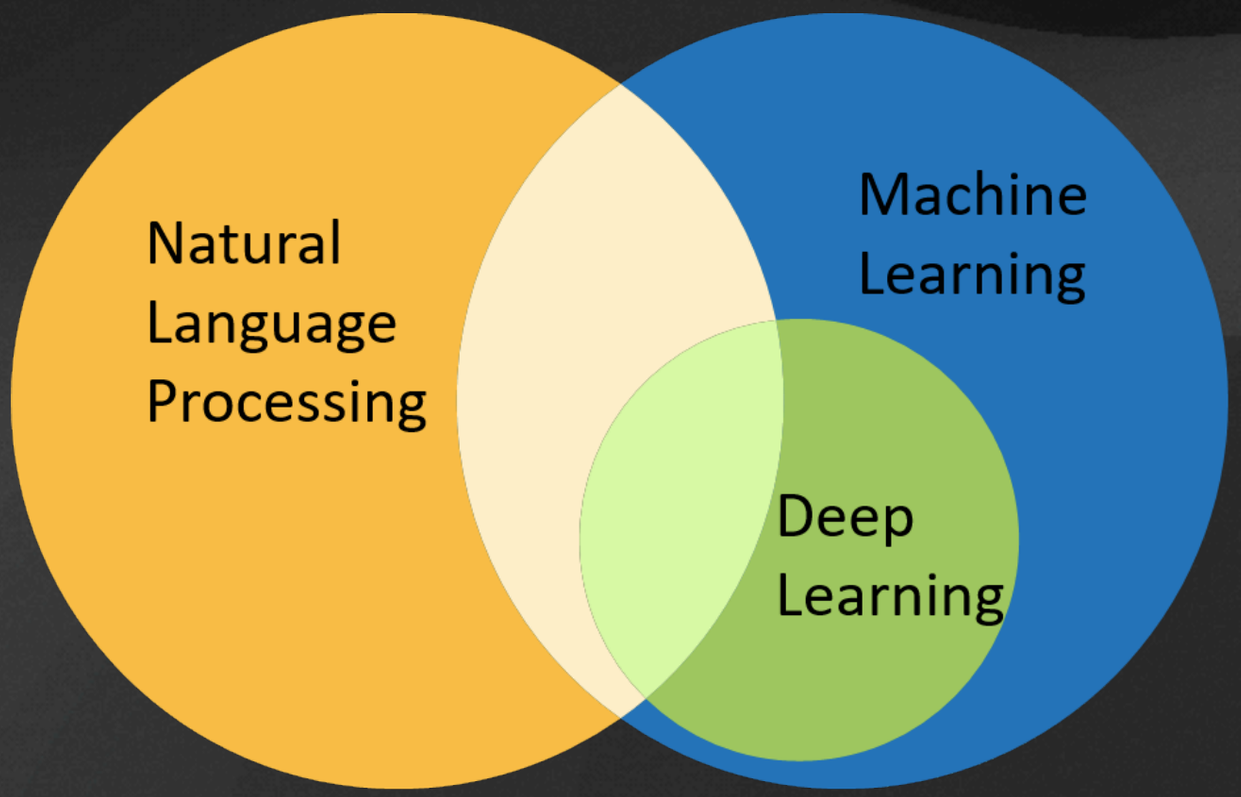


Word2Vec in Python with Gensim Library

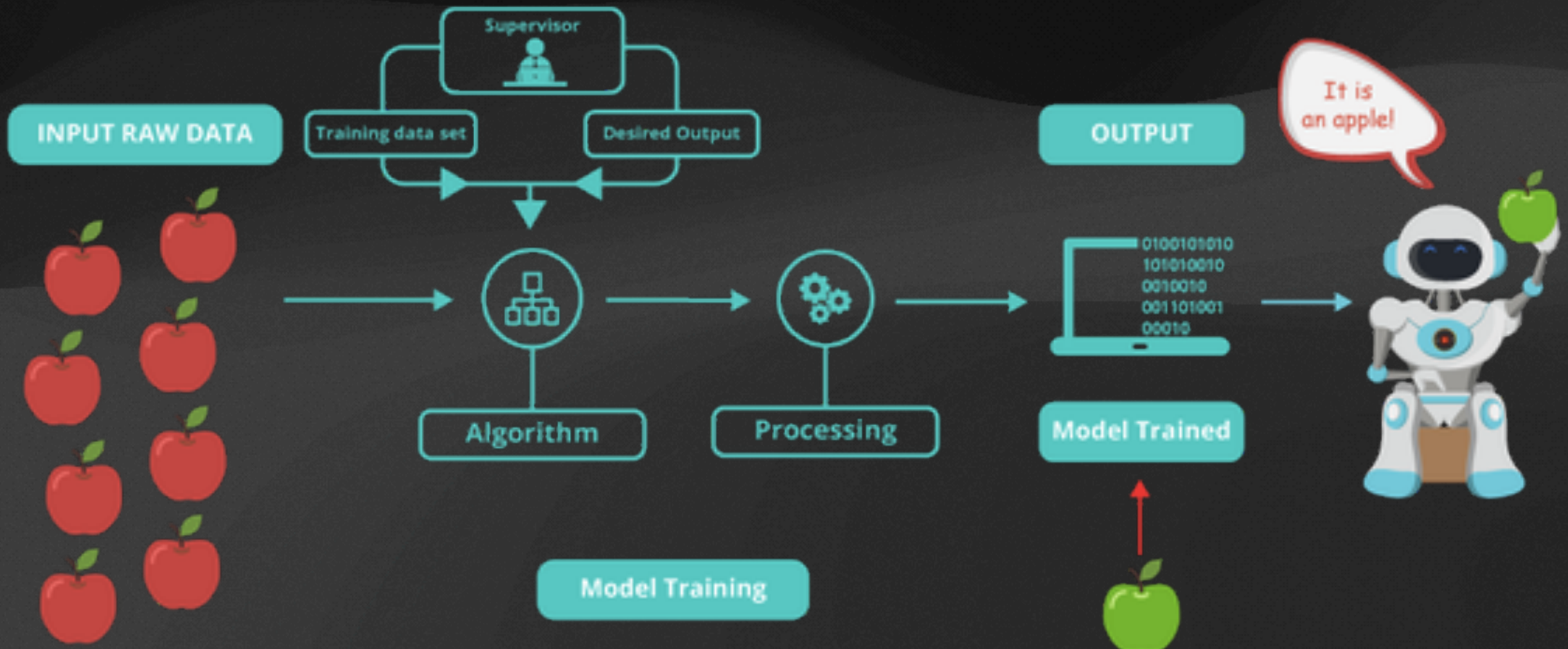
Let’s Create a Simple ChatBot

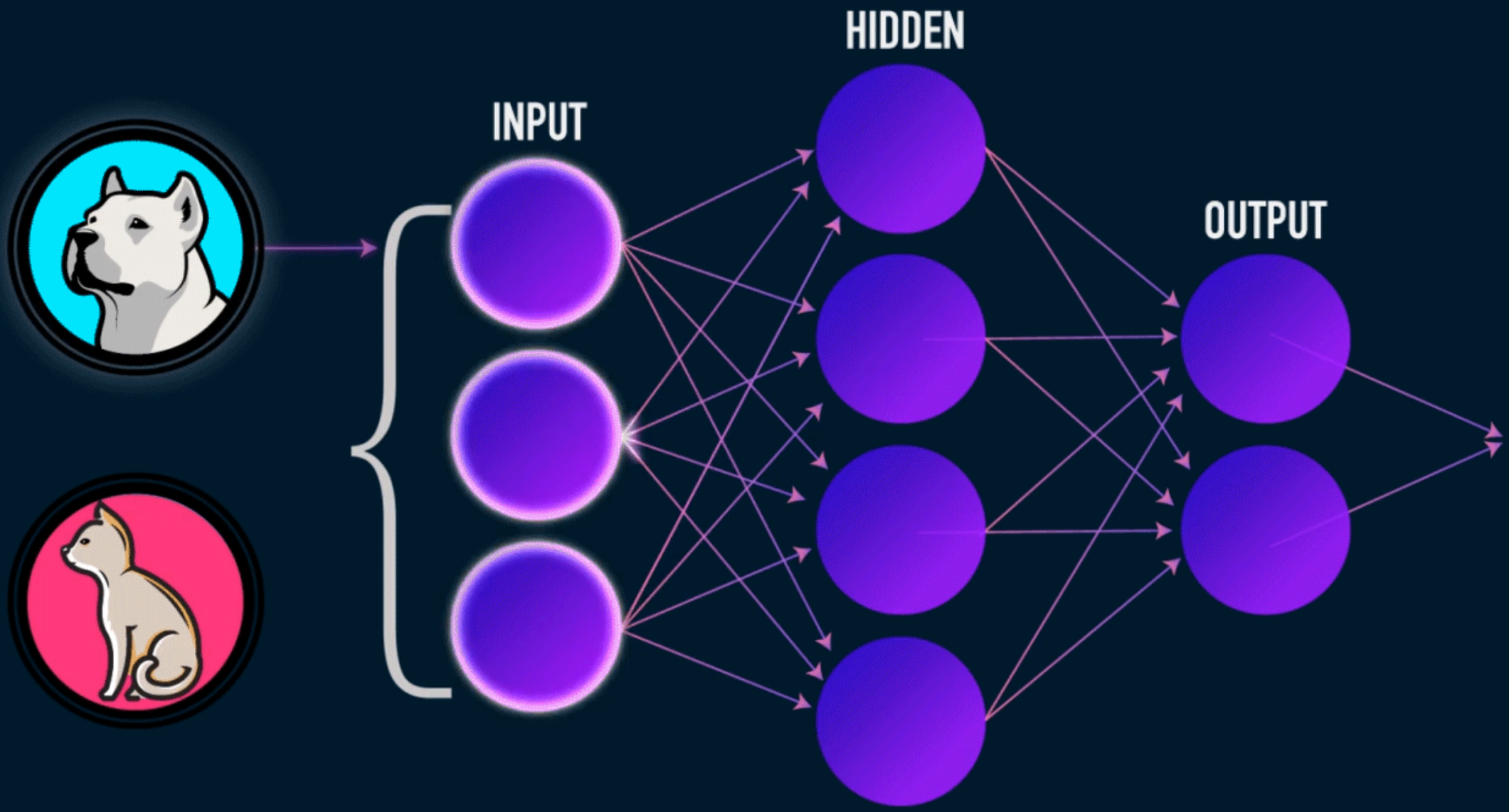
- Creating a JSON file (intents, patterns, responses)
- Processing data : Each intent is tokenized into words and the patterns and their associated tags are added to their respective lists.
- Create Bag of words
- Designing a neural network model

Introduction to Deep Learning



Supervised Learning




HIDDEN
INPUT
OUTPUT

- A neural network is a type of machine learning model based on a number of small mathematical functions called neurons. Like the neurons in a human brain, they are the lowest level of computation.

- Each neuron is a simple mathematical function that calculates an output based on some input. The power of the neural network, however, comes from the connections between the neurons.

- Each neuron is connected to some of its peers, and the strength of each connection is quantified through a numerical weight. They determine the degree to which the output of one neuron will be taken into account as an input to a following neuron.

What is TensorFlow?

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning.

TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.

- **Input Layer**

  The input layer accepts large volumes of data as input to build the neural network. The data can be in the form of text, image, audio, etc.

- **Hidden Layer**

  This layer processes data by performing complex computations and carries out feature extraction. As part of the training, these layers have weights and biases that are continuously updated until the training process is complete. Each neuron has multiple weights and one bias. After computation, the values are passed to the output layer.

- **Output Layer**

  The output layer generates predicted output by applying suitable activation functions. The output can be in the form of numeric or categorical values.

- **Sequential model:** It is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

- Dense layer: The in neural networks is the one that executes matrix-vector multiplication.

- The matrix parameters are retrieved by updating and training using the backpropagation methodology

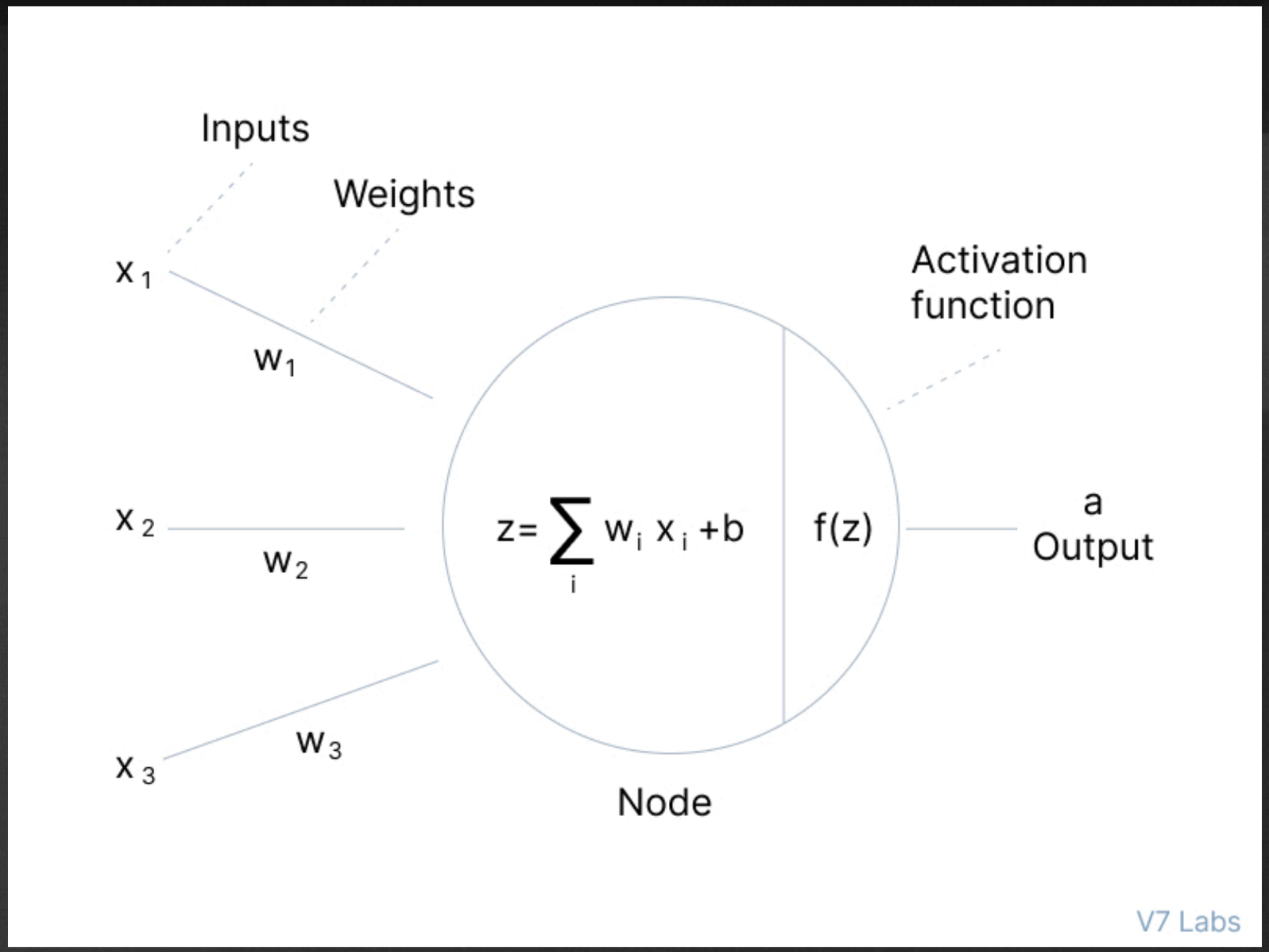
- The final result of the dense layer is the vector of n dimensions

Units: This helps us represent the dimensions required in the output space and should be specified using any positive integer value.

Input shape: In Keras, the input layer itself is not a layer, but a tensor. It's the starting tensor send to the first hidden layer. This tensor must have the same shape as your training data.

Neural Network Activation Function

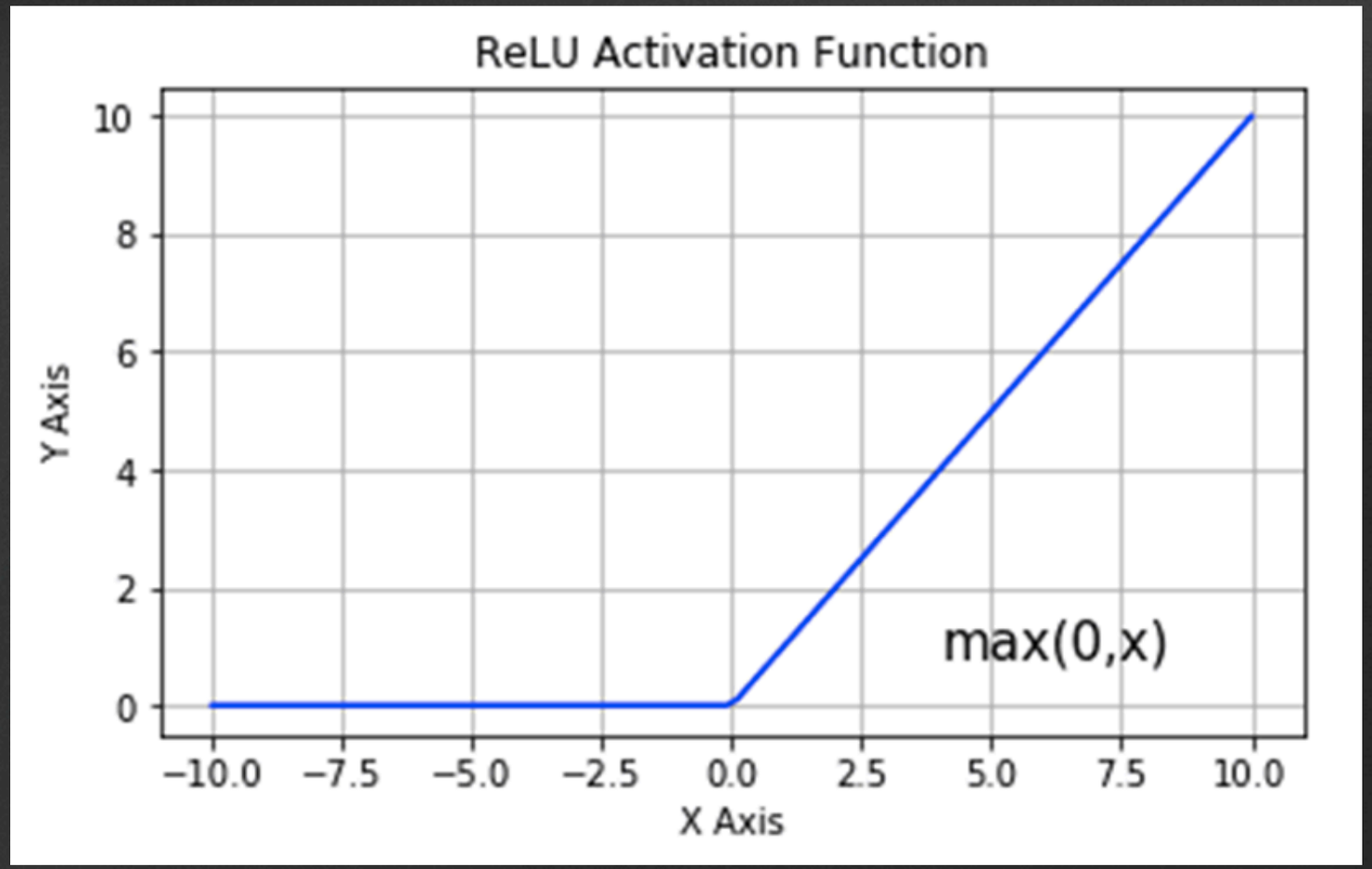
An Activation Function decides whether a neuron should be activated or not. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell



ReLU (Rectified Linear Unit)

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

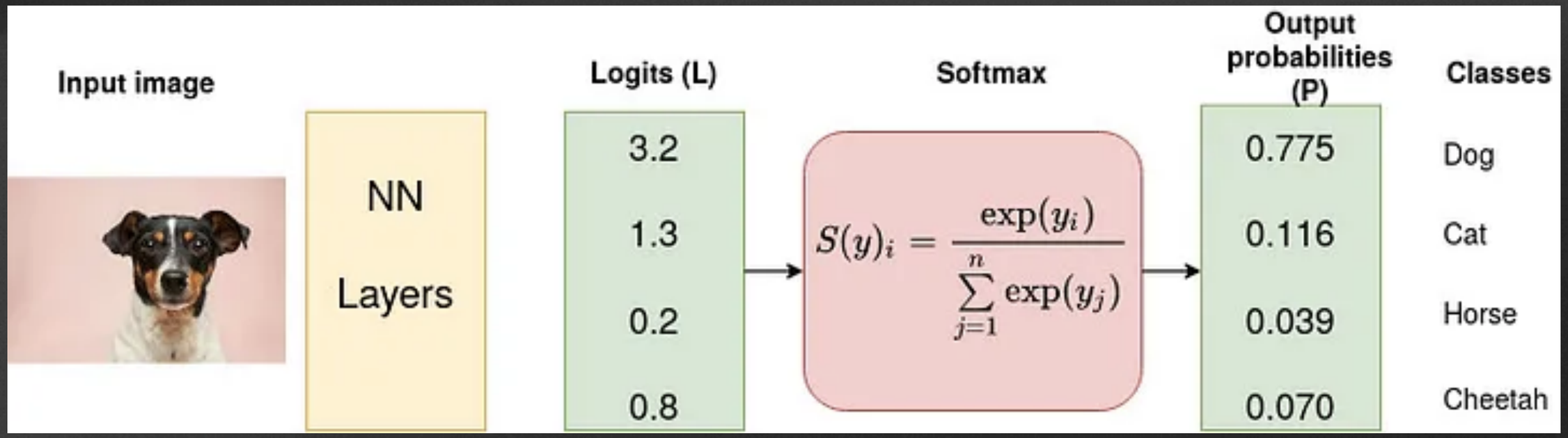
The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.



Softmax

Softmax is an activation function that scales numbers/logits into probabilities.

It is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes



Dropout

It is used to fix the over-fitting issue. Input data may have some of the unwanted data, usually called as Noise. Dropout will try to remove the noise data and thus prevent the model from over-fitting.

Optimizers in Tensorflow

Optimizers are techniques or algorithms used to decrease loss (an error) by tuning various parameters and weights, hence minimizing the loss function, providing better accuracy of model faster.

Create a User Input

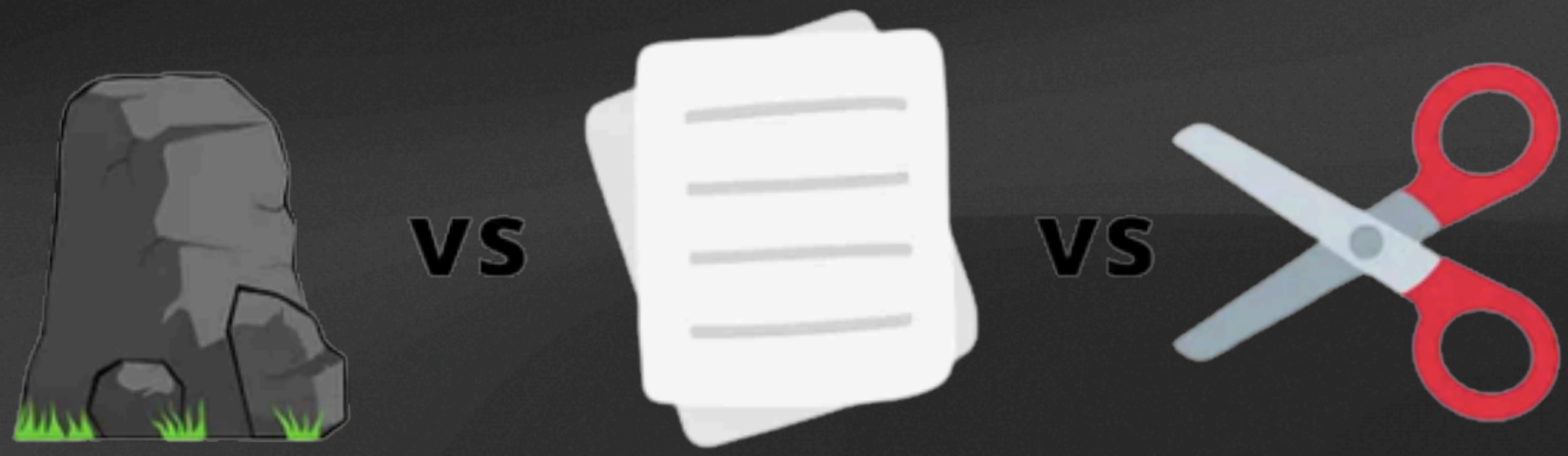
The user will be able to enter a query in a while loop, which will then be cleaned. Next, we use our *bag of words* model to convert our text into numerical values and make a prediction about which tag in our intent the features most closely represent us:


RASA
NLU

What is Rasa?

- RASA is an essential open-source machine learning framework used to build customized AI chatbots. It can be integrated with websites, Facebook, Telegram, Whatsapp, Slack, Discord, etc.
- RASA has two main components: RASA NLU and RASA CORE.
- RASA NLU interprets the user input and extracts entities and intent with the help of various pipelines. It then converts the input into a dictionary that includes the original text, intent, and entities identified, which is then sent to RASA CORE.
- RASA CORE is responsible for the chatbot's response. It selects the appropriate response as per the user input and then sends it back as a chatbot response.
- Basic Terms of Chatbot Development
  - Query: User's message or user input to the chatbot.
  - Response/Action: Chatbots reply or the action, taken by it as per the user input.
  - Intent: The user's primary goal or the intention behind the input or message.
  - Entity: Any piece of important information gathered from the user input. (nouns in the user input)

Create a Rock-Paper-Scissors game with Chatbot



Assistant preview

Hello. How can I help you?

let's

How to Install Rasa:

- Creating VirtualEnv
- python3.8 -m venv rasa_env
- source rasa_env/bin/activate
- Install rasa3
- pip install rasa==3
- rasa init

- **__init__.py:** An empty file that helps python to locate your actions.
- actions.py: This file is used for creating custom actions. In case you want to call an external server or fetch external API data, you can define your actions here.
- **config.yml**: This file defines the configuration of the NLU and core model. If you are using any model outside the NLU model, you have to define the pipeline here.
- **credentials.yml**: This file is used to store credentials for connecting to external services such as Facebook Messenger, Slack, etc
- Data folder consists of :
    - nlu.yml: In this file, we define our intents (what the user could ask the bot to do?). These intents are then used in training the NLU model.
    - rules.yml: define a short conversation path that should always be followed
    - stories.yml: Stories are the sample conversation between a user and bot in the form of intents, responses and actions. Rasa stories are a form of training data used to train the Rasa's dialogue management models.
- Domain.yml is the main file, represents the universe in which the model works. It lists all intents, entities, slots, responses, forms, and actions that your Bot uses and specifies a configuration for conversation sessions.
- **endpoints.yml**: This defines the details for connecting channels like Slack, FB messenger, etc. for storing chats data in the online databases like Redis, etc.
- Models folder contains the trained models.
- Tests folder contains tests to evaluate that your Bot behaves as expected.

YAML Ain’t Markup Language (YAML)

```yaml
version: "3.1"

nlu:
- intent: greet
  examples: |
    - Hey
    - Hi
    - hey there [Sara](name)

- intent: faq/language
  examples: |
    - What language do you speak?
    - Do you only handle english?

stories:
- story: greet and faq
  steps:
  - intent: greet
  - action: utter_greet
  - intent: faq
  - action: utter_faq

rules:
- rule: Greet user
  steps:
  - intent: greet
  - action: utter_greet
```

Let’s Start With stories

- Stories are a type of training data used to train your assistant's dialogue management model. Stories can be used to train models that are able to generalize to unseen conversation paths.

- A story is a representation of a conversation between a user and an AI assistant, converted into a specific format where user inputs are expressed as intents (and entities when necessary), while the assistant's responses and actions are expressed as action names.

- All actions executed by the bot are specified with the action: key followed by the name of the action.

- two types of actions:

  - Bot responses are actions that send a message to a user without running any custom code or returning events. These responses can be defined directly in the domain file under the *responses* key. start with utter_

  - Custom actions: It can run any code you want, including API calls, database queries etc. They can turn on the lights, add an event to a calendar, check a user's bank balance, or anything else you can imagine. start with action_

Story 1(normal):

1. User: Greeting(Hi/Hello/Good morning)
2. Bot : Type 'rock', 'paper' or 'scissors' to play.
3. User: Rock (choice)
4. Bot: Paper(choose random choice from a list[rock , paper , scissors])
5. Bot: play again?

```yaml
- story: play rock paper scissors
  steps:
  - intent: greet
  - action: utter_play
  - intent: inform
  - action: action_play_rps
  - action: utter_play_again
```

- Story 2 (if user type yes) :

```
- story: play again
  steps:
  - action: utter_play_again
  - intent: affirm
  - action: utter_play

- story: don't play again
  steps:
  - action: utter_play_again
  - intent: deny
  - action: utter_goodbye
```

NLU Training Data(NLU.yml)

This usually includes the user's intent and any entities their message contains.

1. Intent: greet
    - Hi/Hello/Good Morning/hey dude
2. Intent: goodbye
    - Good bye/bye/c u/good night
3. Intent: Affirm
    - Yes /of course/indeed
4. Intent: Deny
    - No/never/no way
5. Intent: bot_challeng
    - are you a bot?/ are you a human?
6. Intent : inform
    - - [rock](choice)
    - - [paper](choice)
    - - [scissors](choice)

```
- intent: inform
  examples: |
    - [rock](choice)
    - [paper](choice)
    - [scissors](choice)
```

Domain

The domain defines the universe in which your assistant operates. It specifies the intents, entities, slots, responses, forms, and actions your bot should know about. It also defines a configuration for conversation sessions.

1. Intents: The intents key in your domain file lists all intents used in your NLU data

2. Entities: The entities section lists all entities

```yaml
intents:
  - greet
  - inform
  - goodbye
  - affirm
  - deny
  - bot_challenge

entities:
  - choice
```

Slot

Slo
can
city)

```yaml
slots:
  choice:
    type: categorical
    values:
      - rock
      - paper
      - scissors
mappings:
- type: from_entity
  entity: choice
```

which
home

l values.

cted

Responses

Responses are actions that send a message to a user without running any custom code or returning events. These responses can be defined directly in the domain file under the **responses** key and can include rich content such as buttons and attachments.

```yaml
responses:
  utter_play:
  - text: "Type 'rock', 'paper' or 'scissors' to play."

  utter_play_again:
  - text: "Do you want play again?"

  utter_goodbye:
  - text: "Bye"

  utter_iamabot:
  - text: "I am a bot, powered by Rasa."
```

Actions

Actions are the things your bot can actually do.

For example, an action could:

- respond to a user,
- make an external API call,
- query a database, or

just about anything!

All custom actions should be listed in your domain, except responses which need not be listed under actions: as they are already listed under *responses*:.

```
actions:
  - action_play_rps
```

Session configuration

A conversation session represents the dialogue between the assistant and the user.

- session_expiration_time defines the time of inactivity in minutes after which a new session will begin.
- carry_over_slots_to_new_session determines whether existing set slots should be carried over to new sessions.

Rules

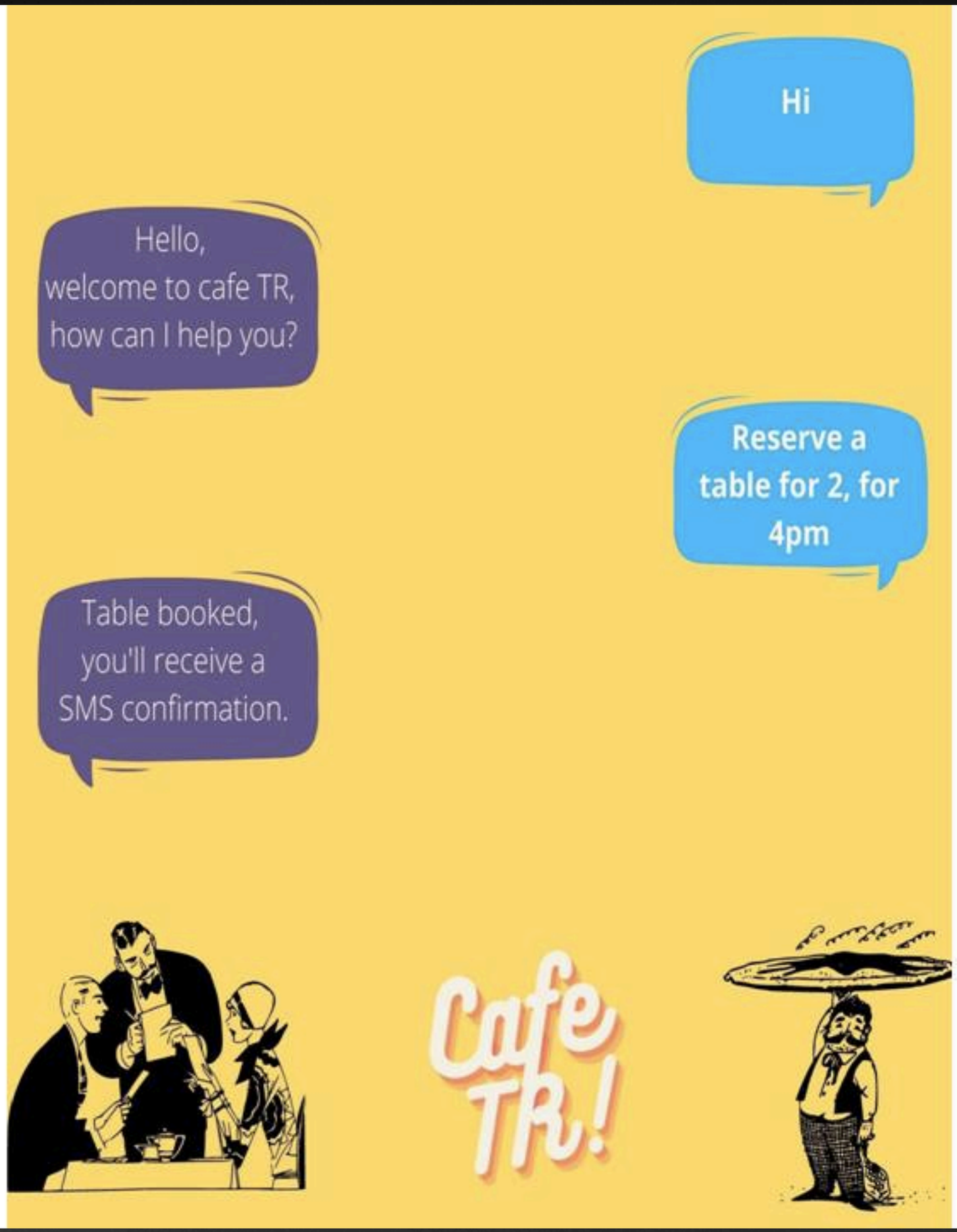
Rules are a type of training data used to train your assistant's dialogue management model. Rules describe short pieces of conversations that should always follow the same path.

Custom Actions

A custom action is an action that can run any code you want. This can be used to make an API call, or to query a database for example.

Response : Greet

Action :
Table reservation



Intent : Greet

Intent: table reservation
Entity: number of people 2, time 4pm

Let’s Make a chatGPT With Custom Knowledge Base

What Is a Large Language Model, the Tech Behind ChatGPT?

- ChatGPT is an artificial intelligence chatbot developed by OpenAI and released in November 2022.
- A Large Language Model Is a Type of Neural Network
- An LLM Uses a Transformer Architecture
- An LLM Predicts Which Word Should Follow the Previous

- install the OpenAI library : It will allow us to interact with ChatGPT through their API

- Install Langchain : LangChain is a framework for developing applications powered by language models.

```
pip install gpt_index
pip install langchain
```

- Gradio allows you to quickly develop a friendly web interface so that you can demo your AI chatbot. It also lets you easily share the chatbot on the internet through a shareable link.

```
pip install gradio
```

GPT INDEX: GPT-Index is a powerful tool that allows you to create a chatbot based on the data feed by you. With GPT-Index, you don't need to be an expert in NLP or machine learning. You simply need to provide the data you want the chatbot to use, and GPT-Index will take care of the rest

Create the function (construct_index)

This is known as embedding, and it must be repeated whenever fresh data is received.

SimpleDirectoryReader(load data):

You can get data from specific URL or you can download data file and give the path of that file.

GPT-Index supports a wide range of data formats, including txt, CSV, JSON, and Excel.

You can also connect GPT-Index to your existing database or API to automatically pull in new data as it becomes available.

PromptHelper: General prompt helper that can help deal with token limitations.The helper can split text.

LLMPredictor: is a wrapper around Langchain's LLMChain that allows easy integration into LlamaIndex.

Write the Ask Function (chatbot)

Finally we will write the function which will ask AI. The function created index file from where it retrieves all file information which we provided in above step

Thank You

- https://www.linkedin.com/in/gettoalexjolly/
- https://github.com/alexjolly28