Studenti

Smahi Hichem

Data Analytics 21/22

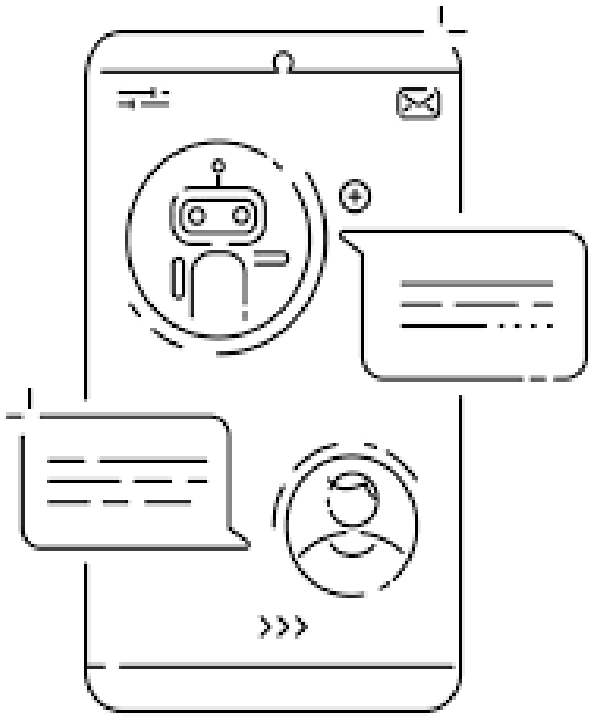
Prof.

Massimo Callisto De Donato

The objective

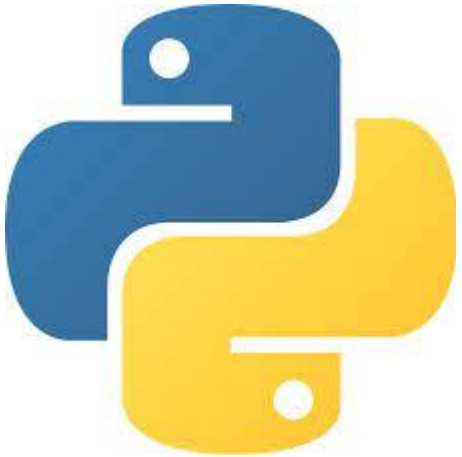
The objective of this project is to define a prototype in python that is able to analyse in real time sentiment about users in chatbot interactions.

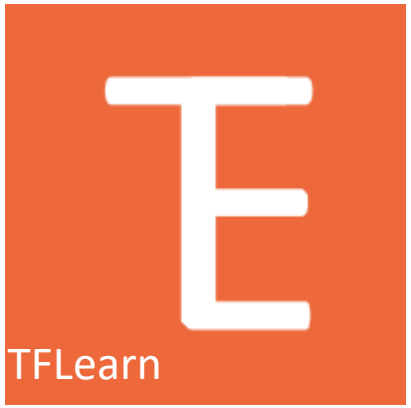
Plan



- Creating chatbot
- Integration of sentiment analysis
- Visualization analysis

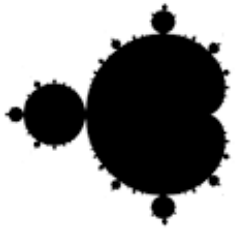
Tools



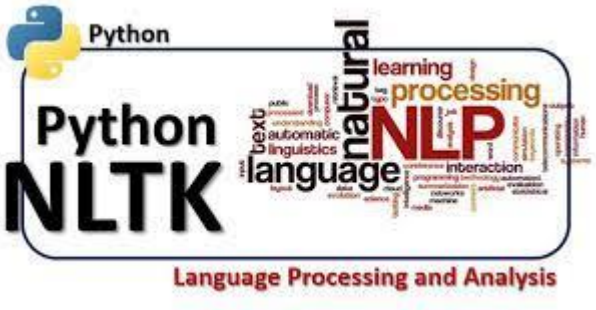

TFLearn


Flask


Development
HTML
C++
PHP


TextBlob


elastic


Python
Python
NLTK
natural
learning
processing
NLP
text
automatic
linguistics
language
interaction
Language Processing and Analysis

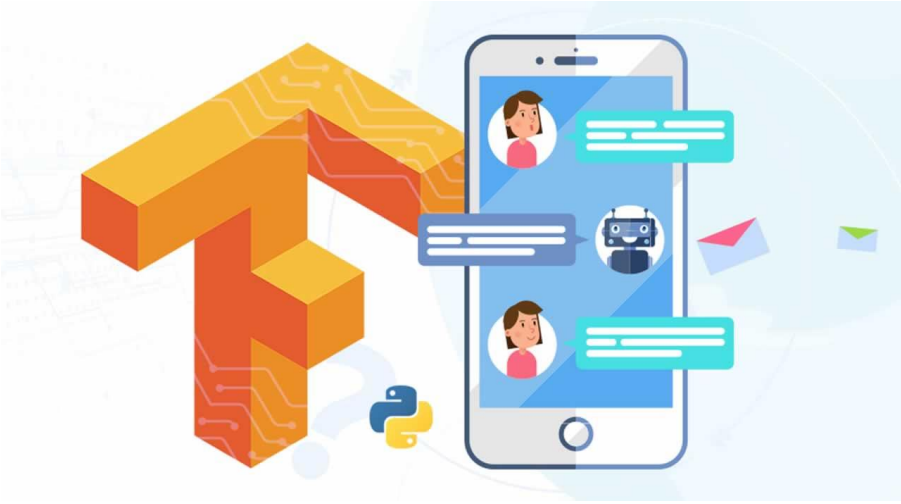

TensorFlow


{JSON}

Creating chatbot

What is Tensorflow Chatbot?

By stimulating conversations in a natural language with a computer program called Dialogflow & Tensorflow Chatbot, they help users create the happiest lives.



How Do I Make An Ai Chatbot In Python?

- Import and load the data file.
- Preprocess data.
- Create training and testing data.
- Build the model.


```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day", "Whats up" , "Good morning" , "Good evening" , "hello" , "hey" , "what's up"],
         "responses": ["Hello!", "Good to see you!", "Hi there, how can I help?"],
         "context_set": ""
        },
        {"tag": "goodbye",
         "patterns": ["cya", "See you later", "Goodbye", "I am Leaving", "Have a Good day","bye","i have to go","gotta go"],
         "responses": ["Sad to see you go :(", "Talk to you later", "Goodbye!"],
         "context_set": ""
        },
        {"tag": "age",
         "patterns": ["how old", "how old are you", "what is your age", "how old are you", "age?"],
         "responses": ["I am old and wise", "trying to stay fit and young!"],
         "context_set": ""
        },
        {"tag": "name",
         "patterns": ["what is your name", "what should I call you", "whats your name?", "who are you?"],
         "responses": ["You can call me Sally.", "I'm Sally!", "I'm Sally."],
         "context_set": ""
        },
        {"tag": "help",
         "patterns": ["Id like to ask something", "what can you do", "can you help me?", "can i tell you something"],
         "responses": ["I'm here to help you!", "I can help you with whatever you tell me to."],
         "context_set": ""
        }
    ]
}
```

```python
import nltk
nltk.download('punkt')
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy
import tensorflow as tf
import tflearn
import random
import json
import pickle

from textblob import TextBlob

from time import sleep

with open("intents.json") as file:
    data = json.load(file)

try:
    with open("data.pickle", "rb") as f:
        words, labels, training, output = pickle.load(f)
```

1

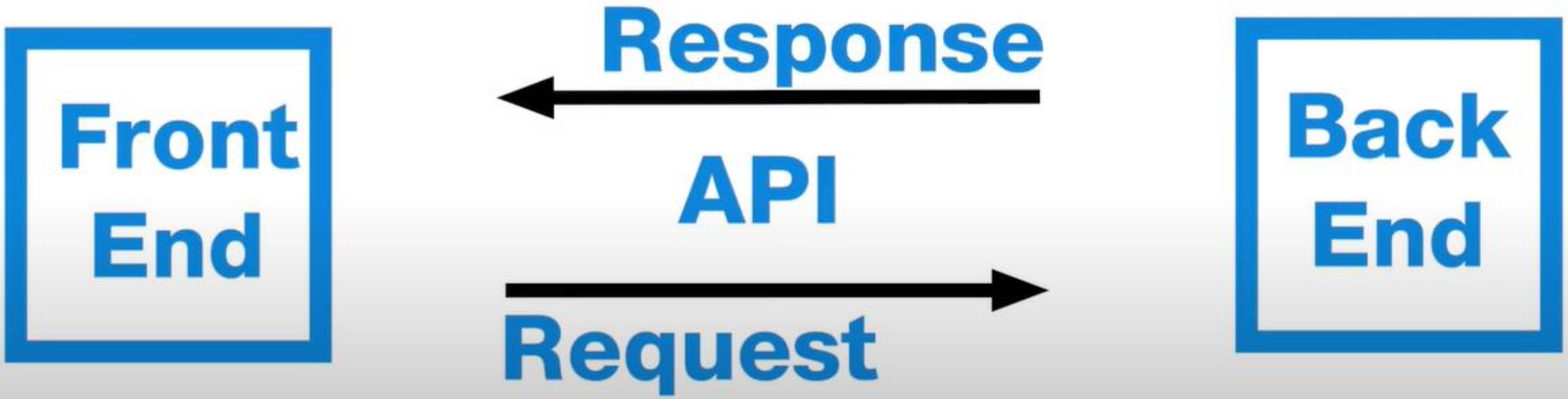
```python
    with open("data.pickle", "wb") as f:
        pickle.dump((words, labels, training, output), f)

try:
    model.load('model.tflearn')
except:
    tf.compat.v1.reset_default_graph()

    net = tflearn.input_data(shape=[None, len(training[0])])
    net = tflearn.fully_connected(net, 8)
    net = tflearn.fully_connected(net, 8)
    net = tflearn.fully_connected(net, len(output[0]), activation='softmax')
    net = tflearn.regression(net)

    model = tflearn.DNN(net)

    model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
    model.save("model.tflearn")

def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

2

```python
results = model.predict([bag_of_words(inp, words)])[0]
results_index = numpy.argmax(results)
tag = labels[results_index]
if results[results_index] > 0.8:
    for tg in data["intents"]:
        if tg['tag'] == tag:
            responses = tg['responses']
    sleep(3)
    Bot = random.choice(responses)
    return(Bot)
else:
    return("I don't understand!")
```

3

Application Programming Interface



```python
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def home():
    return render_template("index.html")

@app.route("/get")
def get_bot_reponse():
    userText = request.args.get('msg')
    return str(process(userText))

if __name__ == "__main__":
    app.run(debug=True)
```

Administrateur : Anaconda Prompt (miniconda3) - python main.py

```
←[A←[ATraining Step: 3994  | total loss: ←[1m←[32m0.04702←[0m←[0m | time: 0.006s
| Adam | epoch: 999 | loss: 0.04702 - acc: 0.9995 -- iter: 16/32
←[A←[ATraining Step: 3995  | total loss: ←[1m←[32m0.04717←[0m←[0m | time: 0.007s
| Adam | epoch: 999 | loss: 0.04717 - acc: 0.9996 -- iter: 24/32
←[A←[ATraining Step: 3996  | total loss: ←[1m←[32m0.04729←[0m←[0m | time: 0.008s
| Adam | epoch: 999 | loss: 0.04729 - acc: 0.9996 -- iter: 32/32
--
Training Step: 3997  | total loss: ←[1m←[32m0.04615←[0m←[0m | time: 0.001s
| Adam | epoch: 1000 | loss: 0.04615 - acc: 0.9997 -- iter: 08/32
←[A←[ATraining Step: 3998  | total loss: ←[1m←[32m0.04739←[0m←[0m | time: 0.003s
| Adam | epoch: 1000 | loss: 0.04739 - acc: 0.9997 -- iter: 16/32
←[A←[ATraining Step: 3999  | total loss: ←[1m←[32m0.04588←[0m←[0m | time: 0.005s
| Adam | epoch: 1000 | loss: 0.04588 - acc: 0.9997 -- iter: 24/32
←[A←[ATraining Step: 4000  | total loss: ←[1m←[32m0.04638←[0m←[0m | time: 0.006s
| Adam | epoch: 1000 | loss: 0.04638 - acc: 0.9998 -- iter: 32/32
--
 * Debugger is active!
 * Debugger PIN: 531-946-015
127.0.0.1 - - [30/Apr/2022 15:22:21] "GET / HTTP/1.1" 200 -
hi , I am very proud to study in UNICAM
{'message': 'hi , I am very proud to study in UNICAM', 'polarity': 1.0, 'subjectivity': 1.0}
127.0.0.1 - - [30/Apr/2022 15:23:11] "GET /get?msg=hi%20,%20I%20am%20very%20proud%20to%20study%20in%20UNICAM HTTP/1.1" 2
00 -
this is good day
{'message': 'this is good day', 'polarity': 0.7, 'subjectivity': 0.6000000000000001}
127.0.0.1 - - [30/Apr/2022 15:23:37] "GET /get?msg=this%20is%20good%20day HTTP/1.1" 200 -
how old are you my bot
{'message': 'how old are you my bot', 'polarity': 0.1, 'subjectivity': 0.2}
127.0.0.1 - - [30/Apr/2022 15:24:14] "GET /get?msg=how%20old%20are%20you%20my%20bot HTTP/1.1" 200 -
```

127.0.0.1:5000

🤦ChatBot

hi , I am very proud to study in UNICAM

I don't understand!

this is good day

Hello!

how old are you my bot

I am old and wise

Type your message here

Why I chose Flask?

Flask is one of the best Python micro framework for deployment of small scale project.

- built-in development server and fast debugger.
- integrated support for unit **testing**.
- RESTful request dispatching.
- Jinja2 **templating**.
- support for secure cookies (client side sessions)
- WSGI 1.0 compliant.
- Unicode based

Why JSON?

It is a text-based way of representing JavaScript object literals, arrays, and scalar data. JSON is relatively easy to read and write, while also easy for software to parse and generate. It is often used **for serializing structured data and exchanging it over a network, typically between a server and web applications**.

Storing JSON data in Elasticsearch

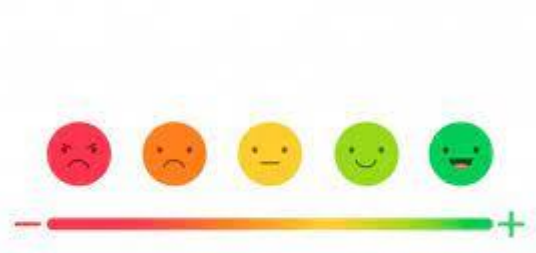
- You can use the re-indexing features Elasticsearch offers. ...
- The update-by-query API only works if there is a _source document.
- You can use Elasticsearch's on-the-fly highlighting feature, which works great for text search.

elastic

```
dat={"message":message ,"polarity":polarity ,"subjectivity":subjectivity}

print(dat)


# function to add to JSON
def write_json(new_data, filename='js.json'):
    with open(filename,'r+') as file:
        # First we load existing data into a dict.
        file_data = json.load(file)
        # Join new_data with file_data inside msg_details
        file_data['mes_details'].append(new_data)
        # Sets file's current position at offset.
        file.seek(0)
        # convert back to json.
        json.dump(file_data, file, indent = 4)

write_json(dat)
```

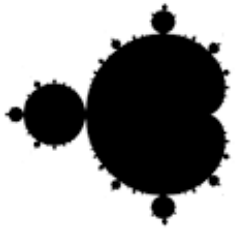
Integration of sentiment analysis

Sentiment Analysis can help us decipher the mood and emotions of general public and gather insightful information regarding the context. Sentiment Analysis is a process of analyzing data and classifying it based on the need of the research.



Sentiment Analysis using TextBlob

TextBlob is a python library for Natural Language Processing (NLP).TextBlob actively used Natural Language ToolKit (NLTK) to achieve its tasks. NLTK is a library which gives an easy access to a lot of lexical resources and allows users to work with categorization, classification and many other tasks. TextBlob is a simple library which supports complex analysis and operations on textual data.

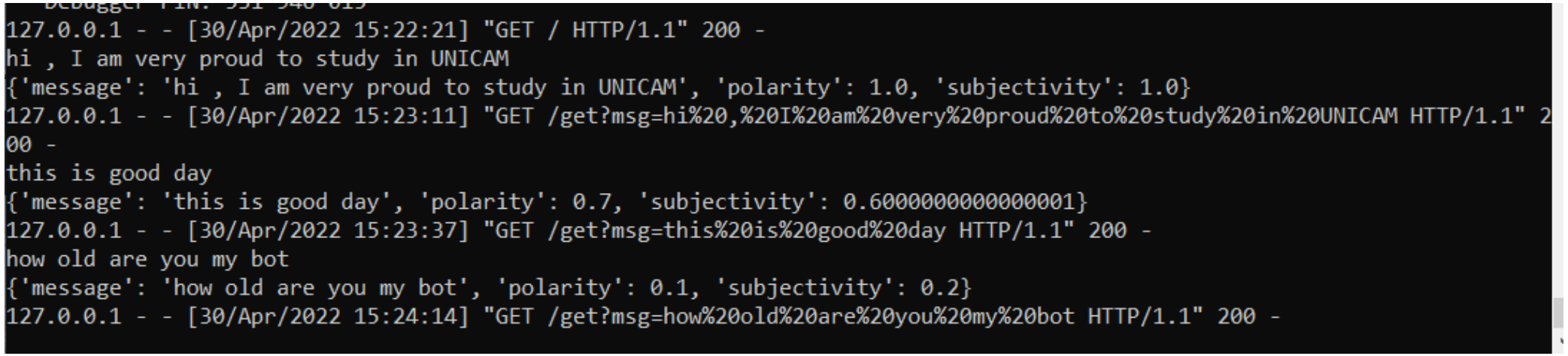


```
mssg=TextBlob(message)
polarity=mssg.sentiment.polarity
subjectivity=mssg.sentiment.subjectivity
dat={"message":message ,"polarity":polarity ,"subjectivity":subjectivity}
```

TextBlob returns **polarity** and **subjectivity** of a sentence.

Polarity lies between [-1,1] , -1 defines a negative sentiment and 1 defines a positive sentiment. Negation words reverse the polarity. TextBlob has semantic labels that help with fine-grained analysis. For example — emoticons, exclamation mark, emojis, etc.

Subjectivity lies between [0,1]. **Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information**.



OUTPUT JSON file


```
{
    "mes_details": [
        {
            "message": "hi",
            "polarity": 0.0,
            "subjectivity": 0.0
        },
        {
            "message": "i m not good",
            "polarity": -0.35,
            "subjectivity": 0.6000000000000001
        },
        {
            "message": "i m good",
            "polarity": 0.7,
            "subjectivity": 0.6000000000000001
        },
        {
            "message": "your age is good",
            "polarity": 0.7,
            "subjectivity": 0.6000000000000001
        },
        {
            "message": "i m not happy , how are you",
            "polarity": -0.4,
            "subjectivity": 1.0
        },
        {
            "message": "i m very proud to study in UNICAM",
            "polarity": 1.0,
            "subjectivity": 1.0
        },
        {
            "message": "I m very good",
            "polarity": 0.9099999999999999,
            "subjectivity": 0.7800000000000001
        },
```

Visualization analysis

What is Elasticsearch?

Initially released in 2010, Elasticsearch (sometimes dubbed ES) is a modern search and analytics engine which is based on Apache Lucene. Completely open source and built with Java, Elasticsearch is a NoSQL database. That means it stores data in an unstructured way and that you cannot use SQL to query it.

This Elasticsearch tutorial could also be considered a NoSQL tutorial. However, unlike most NoSQL databases, Elasticsearch has a strong focus on search capabilities and features — so much so, in fact, that the easiest way to get data from ES is to search for it using the extensive Elasticsearch API.

In the context of data analysis, Elasticsearch is used together with the other components in the ELK Stack, Logstash and Kibana, and plays the role of data indexing and storage.

Kibana

Kibana is a visual interface tool that allows you to explore, visualize, and build a dashboard over the log data massed in Elasticsearch Clusters. Elastic is the company behind Kibana and the two other open source tools - Elasticsearch and Logstash. The Elasticsearch tool serves as the database for document-oriented and semi-structured data. Logstash supports to collect, parse, and store logs for future use. These three tools can work well together and popularly known as ELK Stack or Elastic Stack.

The core feature of Kibana is data querying & analysis. In addition, Kibana's visualization features allow you to visualize data in alternate ways using heat maps, line graphs, histograms, pie charts, and geospatial support. With various methods, you can search the data stored in Elasticsearch for root cause diagnostics.

With Kibana, it is easy to understand big data, and you can quickly build and share dynamic dashboards that frame-out changes to the Elasticsearch query in real-time. This visualization tool is equipped with various options

Visualization analysis Chatbot