

# Inference Deep Learning Model to Web Service



Oleh:  
-Bagaskara  
Pamungkas

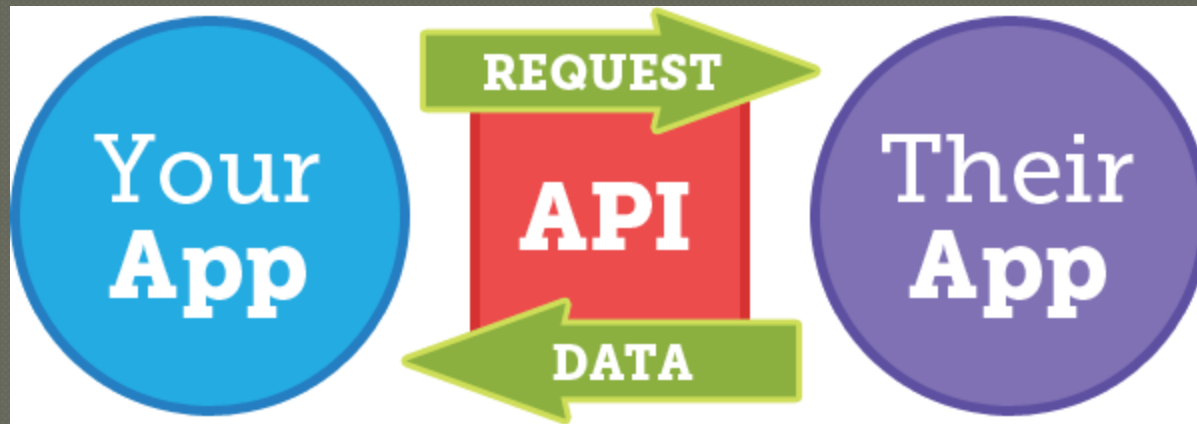
# LINK MATERI DRIVE

---

[shorturl.at/cgyG0](http://shorturl.at/cgyG0)

# APA ITU WEB SERVICE?

---



sumber : <https://qph.ec.quoracdn.net/main-qimg-112d229edc555091db7eb0a3cfc7f7a>

*Web service* adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda

# JENIS-JENIS WEB SERVICE

---

- ◎ - SOAP(Simple Object Access Protocol):

- ◎ adalah sebuah spesifikasi protokol untuk pertukaran pesan/informasi terstruktur dalam implementasi web servis di jaringan komputer. SOAP menggunakan Extensible Markup Language (XML) sebagai format pesannya

- ◎ - REST(Representational State Transfer):

- ◎ Merupakan standar arsitektur berbasis web yang menggunakan protokol HTTP untuk berkomunikasi data

# XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

# JSON

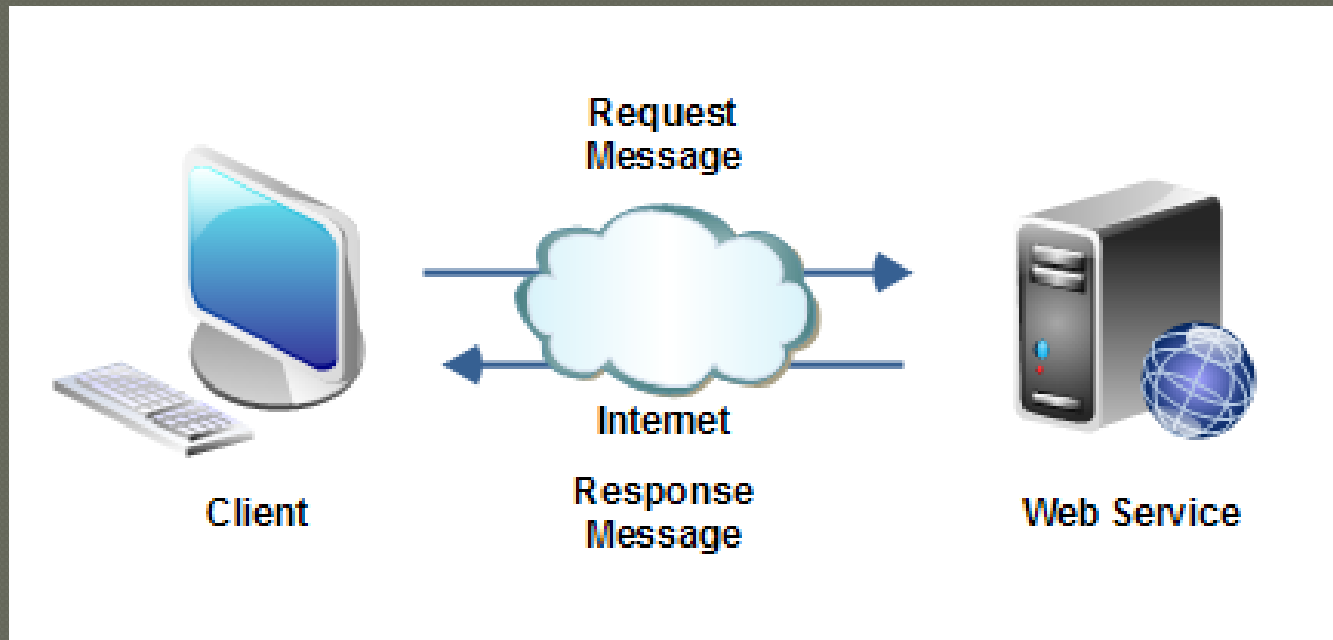
```
{ "empinfo" :
  {
    "employees" : [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```

Uraian	SOAP	REST
(1)	(2)	(3)
Protokol komunikasi	HTTP, HTTPS, SMTP, FTP	HTTP, HTTPS
Penggunaan bandwidth	Dalam jumlah request yang banyak, relatif boros bandwidth. Hal ini karena banyaknya markup dalam penulisan format XML	Relatif hemat bandwidth, karena markup-markup ekstra seperti pada XML tidak dipakai
Tren penggunaan	Banyak mulai beralih ke REST, meski masih tetap ada yang mempertahankan, misalnya untuk integrasi aplikasi ke sistem legasi pada sebuah perusahaan.	Mulai populer, banyak dipakai oleh penyedia web servis terkemuka, seperti twitter, yahoo!, flickr, bloglines, technorati, google, amazon, eBay, dsb
Aturan penulisan	Ketat, mengikuti spesifikasi XML (SOAP v1.2)	Tidak ada spesifikasi khusus
Format respon	XML dengan spesifikasi SOAP. Agak sulit bagi kita untuk membaca langsung dan memahaminya.	XML, JSON, atau format plain teks lainnya. Hal ini memudahkan penerima respon membaca dan memahaminya.
Attachment file	Bisa (karena dapat mengembalikan respon dalam format binary)	Tidak bisa
Sifat web servis pada umumnya	Tertutup, lebih ditujukan untuk vendor atau perusahaan tertentu	Terbuka, bisa diakses siapa saja
Caching web	Relatif sulit	Mudah, karena menggunakan URI
Penggunaan standar	Standar lama (XML, HTTP) dan baru (SOAP) digunakan bersamaan	Standar yang sudah ada, seperti XML dan HTTP
Tool pengembangan	Banyak, baik komersial maupun opensource	Beberapa, karena tidak begitu dibutuhkan
Tool manajemen	Perlu, bahkan kadang harganya mahal	Menggunakan tool yang sudah ada pada sistem jaringan
Ekstensibel	Bisa, banyak ekstensi termasuk standar WS-*	Relatif tidak ekstensibel
Kemudahan implementasi	Mudah jika kita sudah memiliki lingkungan berbasis SOAP	Mudah

Sumber : <http://pusdiklat.bps.go.id/index.php?r=artikel/view&id=206>

# REST API WEB SERVICE

- Metode Umum : GET, POST, PUT, DELETE



### **\*KOMPONEN HTTP REQUEST:**

- HTTP method seperti GET, POST, PUT, DELETE
- URI untuk mengetahui lokasi data di server
- Request Header, berisi metadata seperti Authorization, tipe client dan lain
- Request Body, data yang diberikan client ke server seperti URI params

### **\*KOMPONEN HTTP RESPONSE:**

- Response Code, status server terhadap request yang diminta seperti 200, 500, 404 dan lainnya.
- Response Header yang berisi meta data seperti content type, cache tag dan yang lainnya.
- Response Body, data/resource yang diberikan oleh server baik itu berupa text, json ataupun xml



# FLASK FRAMEWORK

---



**Flask** adalah kerangka kerja aplikasi web mikro yang ditulis dalam bahasa pemrograman Python dan berdasarkan Werkzeug toolkit dan template engine Jinja2 berlisensi BSD.

LETS TRY TO CODING

# LANGKAH AWAL





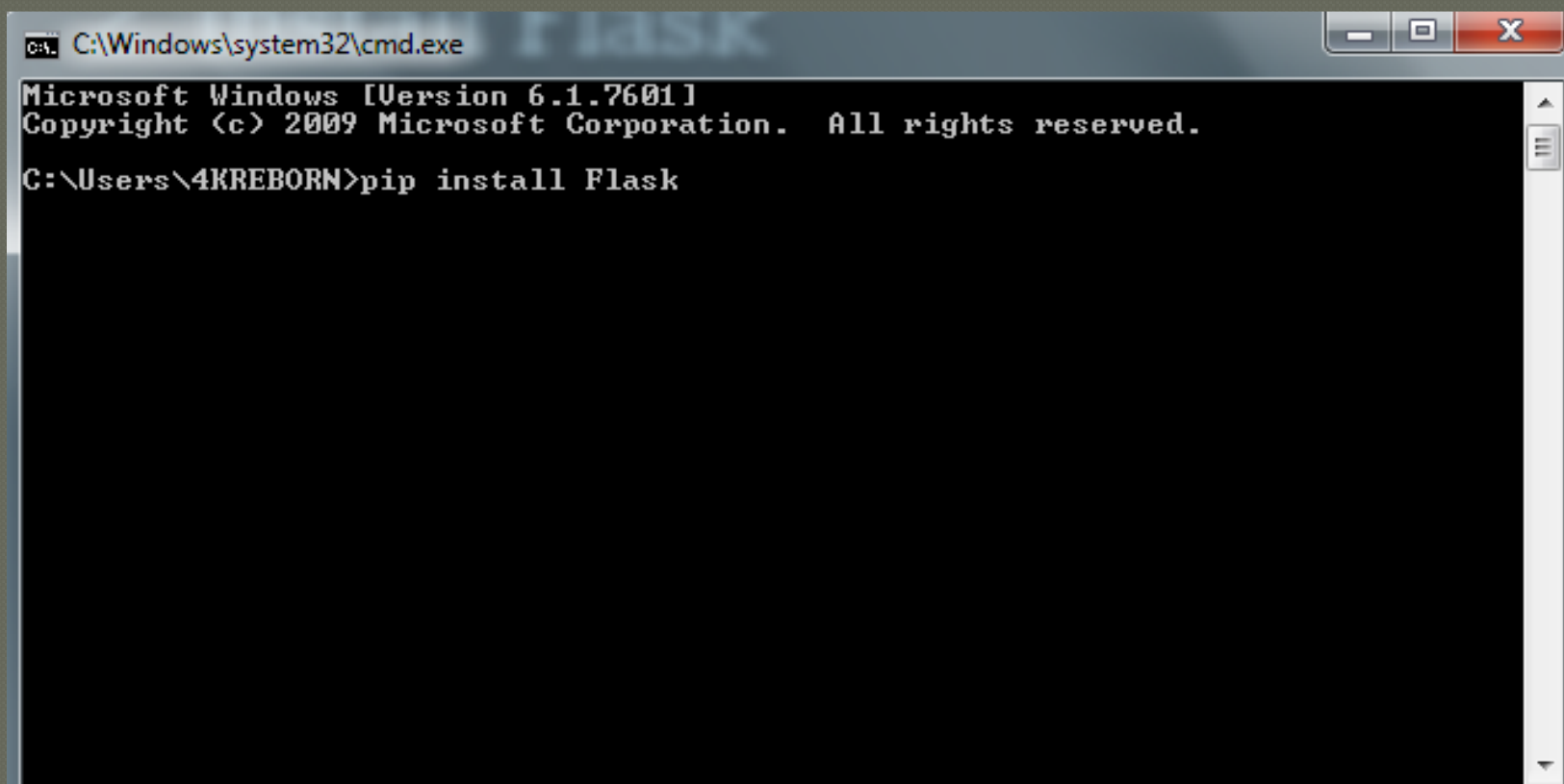
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\4KREBORN>python --version
Python 3.7.3

C:\Users\4KREBORN>
```

# Install Flask

---






A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Windows\system32\cmd.exe" and includes standard minimize, maximize, and close buttons. The command prompt displays the following text:

```
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
C:\Users\4KREBORN>pip install Flask
```

The command prompt is currently at the end of the command line, waiting for further input.

# Buat Folder Baru

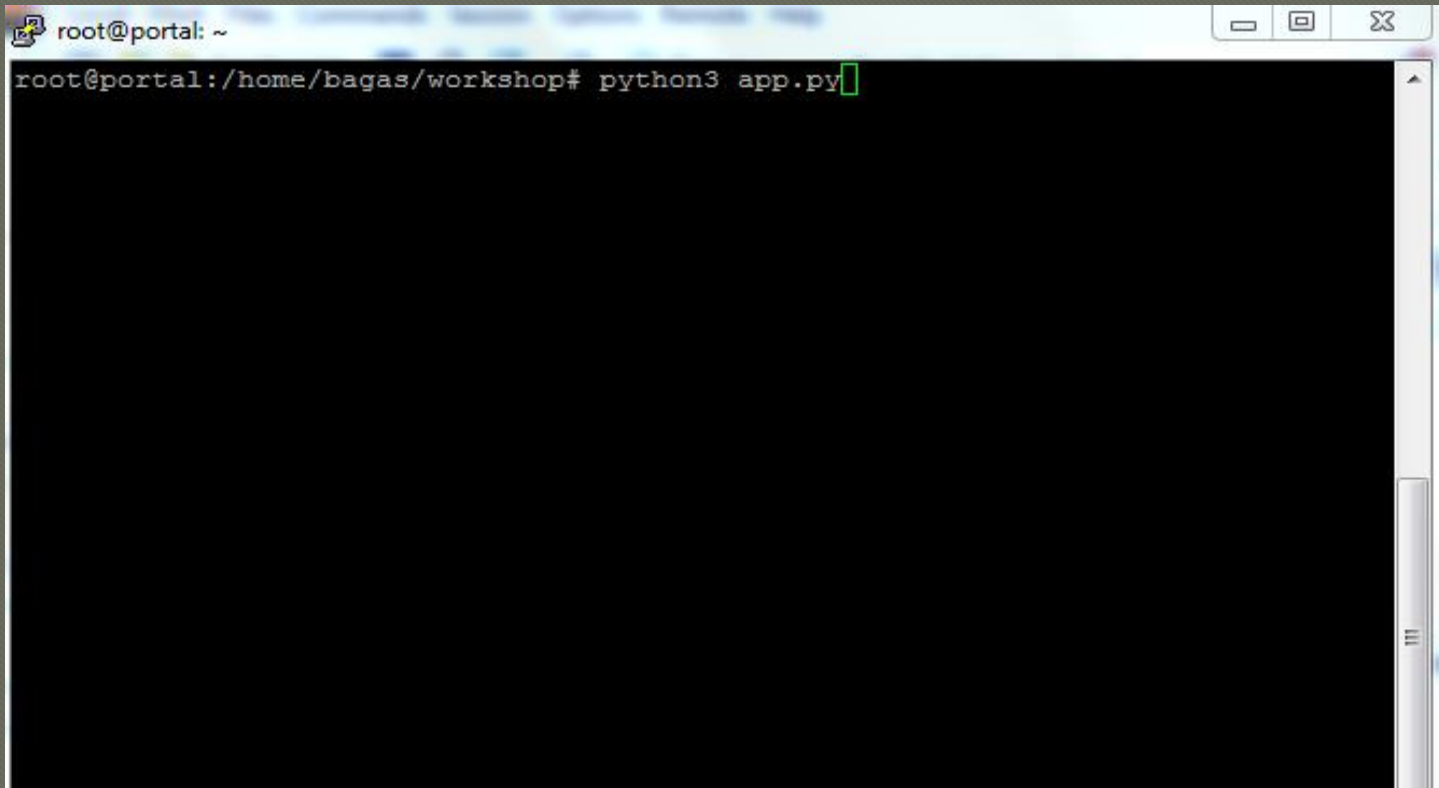
---

Name	Size	Changed
 		7/27/2019 1:04:43 AM
 app.py	1 KB	7/27/2019 1:04:54 AM

## File app.py

```
1
2
3 from flask import Flask
4
5 app = Flask(__name__)
6
7
8 @app.route('/')
9 def hello_world():
10     return 'Hello, World!'
11
12 if __name__ == '__main__':
13     app.run(host="0.0.0.0", port=5150, debug=True)
```

Jalankan Web service dengan command python “namafile”.py

A terminal window with a white title bar and standard Linux window controls (minimize, maximize, close). The terminal text shows the user is root at a machine named portal, in the directory /home/bagas/workshop. The command python3 app.py has been entered, and the cursor is at the end of the line.

```
root@portal: ~  
root@portal:/home/bagas/workshop# python3 app.py
```



# Langkah Berikutnya



# Contoh model yang akan di inference ke Web Service

```
In [1]: 1 import numpy as np
2 from keras.datasets import imdb
3 from keras.models import Sequential
4 from keras.layers import Dense
5 from keras.layers import Flatten
6 from keras.layers.convolutional import Conv1D
7 from keras.layers.convolutional import MaxPooling1D
8 from keras.layers.embeddings import Embedding
9 from keras.preprocessing import sequence
10 from keras.preprocessing.text import one_hot
```

Using TensorFlow backend.

```
In [2]: 1 top_words = 5000
2 max_words = 500
3
4
5 # save np.load
6 np_load_old = np.load
7
8 # modify the default parameters of np.load
9 np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)
10
11 # call load_data with allow_pickle implicitly set to true
12 (X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
13
14 # restore np.load for future normal usage
15 np.load = np_load_old
```

```
In [3]: 1 max_words = 500
2 X_train = sequence.pad_sequences(X_train, maxlen=max_words)
3 X_test = sequence.pad_sequences(X_test, maxlen=max_words)
```





```
In [5]: 1 model = Sequential()
2 model.add(Embedding(top_words, 32, input_length=max_words))
3 model.add(Flatten())
4 model.add(Dense(250, activation='relu'))
5 model.add(Dense(1, activation='sigmoid'))
6 model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
7 #print(model.summary())
```

```
In [7]: 1 #model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5, batch_size=128, verbose=2)
2 scores = model.evaluate(X_test, y_test, verbose=0)
3 print("Accuracy: %.2f%%" % (scores[1]*100))
```

Accuracy: 86.50%

```
In [ ]: 1 model.save("savemodel.h5")
2 model.save_weights("saveweights.h5")
```

Ditaruh didalam satu folder yang sama

Name	Size	Changed
 ..		7/26/2019 12:49:50 AM
 app.py	2 KB	7/27/2019 12:18:08 AM
 savemodel.h5	48,782 KB	7/25/2019 2:53:59 PM
 saveweights.h5	16,267 KB	7/25/2019 2:51:29 PM

## Buat function baru yang bernama predictservice()

```
18 def hello_world():
19     return 'Hello, World!'
20
21 @app.route('/predictservice', methods=['POST'])
22 def predictservice():
23     if request.method == 'POST':
24         K.clear_session()
25         model = load_model('savemodel.h5')
26         model.load_weights('saveweights.h5')
27
28         array = []
29         data = request.json
30         teks = data['teks']
31
32         array.append(teks)
33
34         vocab_size = 5000
35         encoded_teks = [one_hot(t, vocab_size) for t in array]
36
37         max_words = 500
38         encoded_teks = sequence.pad_sequences(encoded_teks, maxlen=max_words)
39
40         hasil = ""
41         predik = model.predict_classes(encoded_teks)
42
43
44         if predik == 0:
45             hasil = "positive"
46
47         if predik == 1:
48             hasil = "negative"
49
50     return jsonify({"hasil": hasil})
51     K.clear_session()
```

## Buat function untuk validasi akurasi model

```
@app.route('/testing',methods=['GET'])
def test():
    if request.method == 'GET':
        K.clear_session()
        model = load_model('savemodel.h5')
        #model.load_weights('saveweights.h5')

        top_words = 5000
        max_words = 500

        # save np.load
        np_load_old = np.load

        # modify the default parameters of np.load
        np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)

        # call load_data with allow_pickle implicitly set to true
        (X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)

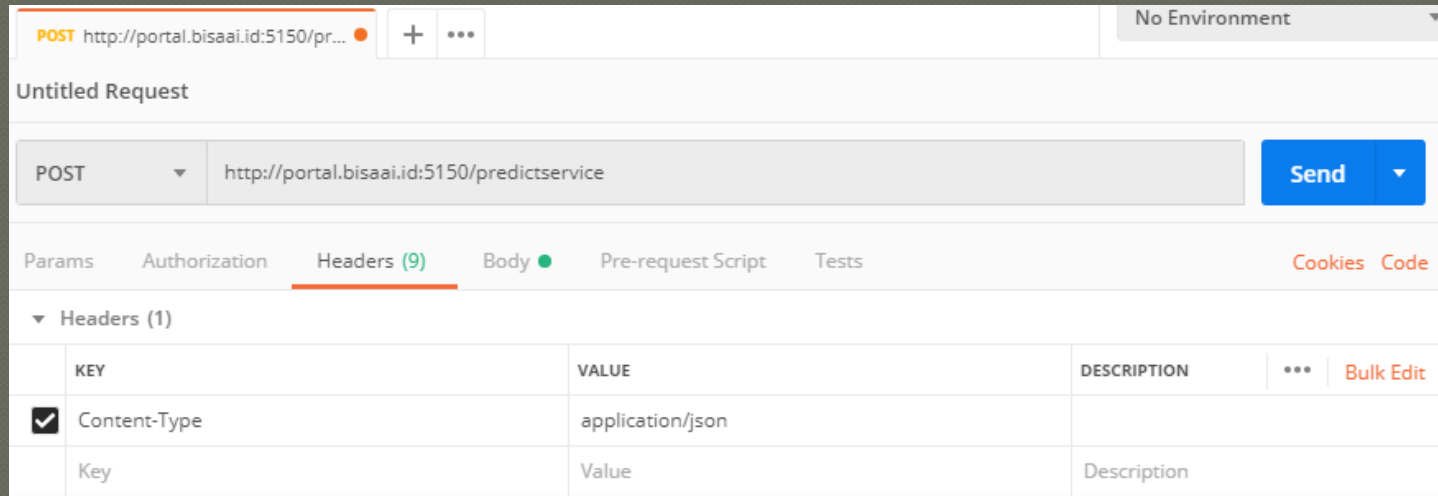
        # restore np.load for future normal usage
        np.load = np_load_old

        max_words = 500
        X_train = sequence.pad_sequences(X_train, maxlen=max_words)
        X_test = sequence.pad_sequences(X_test, maxlen=max_words)

        scores = model.evaluate(X_test, y_test, verbose=0)
        z = "Accuracy: %.2f%%" % (scores[1]*100)
        return jsonify({"hasil":z})
        K.clear_session()
```

# Test Web Service menggunakan Postman

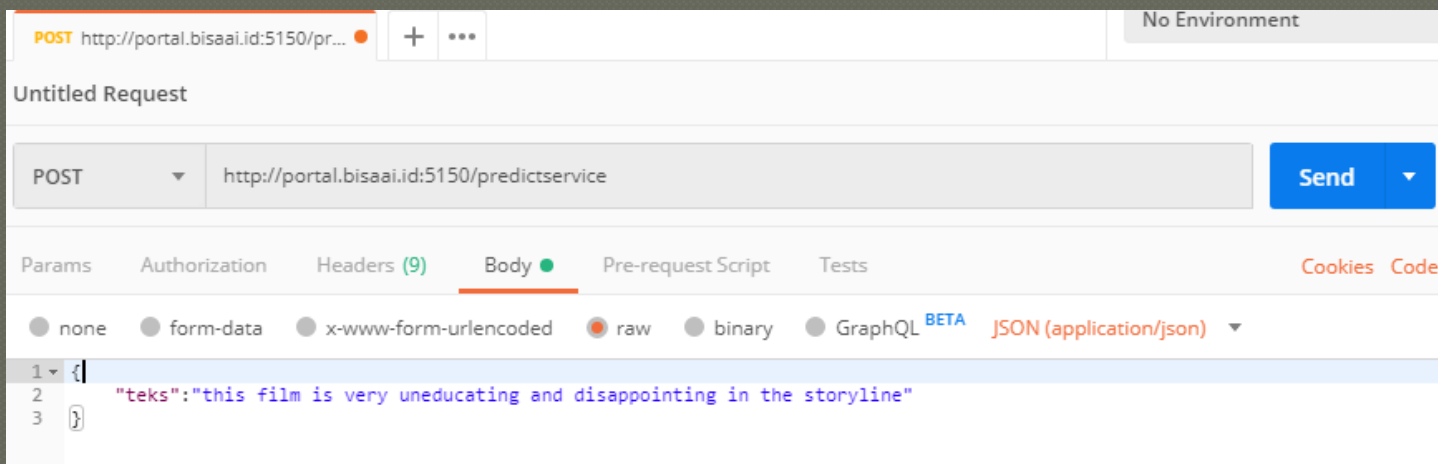
## Header



The screenshot shows the Postman interface for a POST request. The URL is `http://portal.bisaai.id:5150/predictservice`. The Headers tab is selected, showing a table with one header: `Content-Type` with value `application/json`. The table has columns for KEY, VALUE, and DESCRIPTION. There are also buttons for adding new headers (+) and deleting existing ones (...).

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/json	
Key	Value	Description

## Body



The screenshot shows the Postman interface for a POST request. The URL is `http://portal.bisaai.id:5150/predictservice`. The Body tab is selected, showing a JSON body. The body is `{ "teks": "this film is very uneducating and disappointing in the storyline" }`. The body is formatted as JSON (application/json).

```
1 {  
2   "teks": "this film is very uneducating and disappointing in the storyline"  
3 }
```

# Hasil Prediksi

The screenshot displays a REST client interface with a POST request to `http://portal.bisaai.id:5150/predictservice`. The response status is `200 OK` with a time of `1139ms` and a size of `172 B`. The response body, shown in JSON format, contains the prediction result: `"hasil": "negative"`.

POST `http://portal.bisaai.id:5150/pr...` + ... No Environment

POST `http://portal.bisaai.id:5150/predictservice` Send

Body Cookies Headers (4) Test Results Status: 200 OK Time: 1139ms Size: 172 B Save

Pretty Raw Preview JSON ≡

```
1 {  
2   "hasil": "negative"  
3 }
```