MODUL PRAKTIKUM KECERDASAN BUATAN

PERTEMUAN 9 – PEMROSESAN BAHASA ALAMI

Tools : Jupyter notebook
Bahasa Pemrograman : Python

Dalam praktikum kali ini, kita akan belajar tentang bagaimana cara membuat Chatbot sederhana menggunakan Python.

Pemrosesan Bahasa Alami (*Natural Language Processing/ NLP*)

Pemrosesan bahasa alami (*Natural language processing*/NLP) merupakan cabang kecerdasan buatan yang membantu komputer memahami, menafsirkan, dan memanipulasi bahasa manusia. Konsep NLP melibatkan analisis teks dan bahasa, pemahaman konteks, semantik, sintaks, dan tata bahasa. Dengan menggunakan teknik-teknik seperti mesin pencari, pengenalan wicara, serta pemrosesan bahasa alami berbasis aturan, NLP dapat membantu pengguna untuk menghasilkan keluaran bahasa manusia yang diinginkan atau memahami keluaran bahasa manusia dari pengguna lain yang terdiri dari teks, audio, ataupun video. NLP menjadi sangat penting dalam pengembangan teknologi digital dan AI karena membantu meningkatkan kemampuan mesin untuk berinteraksi dengan manusia, memproses dan menganalisa teks yang disajikan, serta mengembangkan aplikasi yang responsif terhadap permintaan bahasa alami. NLP bukanlah ilmu baru, teknologi ini berkembang pesat berkat meningkatnya minat komunikasi manusia-ke-mesin, ditambah ketersediaan big data, komputasi canggih, dan algoritma yang disempurnakan.

Sebagai manusia, kita dapat berbicara dan menulis dalam bahasa Indonesia, Inggris, dll. Namun bahasa alami komputer – yang dikenal sebagai kode mesin atau bahasa mesin – sebagian besar tidak bisa dipahami oleh kebanyakan orang. Pada tingkat perangkat yang paling rendah, komunikasi terjadi bukan dengan kata-kata tetapi melalui jutaan nol dan yang menghasilkan tindakan logis. Saat ini, kita dapat berkata, “Alexa, saya suka lagu ini,” dan perangkat pemutar musik secara otomatis akan mengecilkan volumenya lalu menjawab, “Oke. Peringkat disimpan,” dengan suara yang mirip manusia. Kemudian perangkat tersebut menyesuaikan algoritmanya untuk memainkan lagu itu, pada saat Anda mendengarkan stasiun musik itu di lain waktu.

Mengapa NLP Penting?

Pemrosesan bahasa alami (NLP) sangat penting untuk sepenuhnya menganalisis data teks dan ucapan secara efisien. Teknologi ini dapat menjelajahi berbagai perbedaan dalam dialek, bahasa gaul, dan penyimpangan tata bahasa yang khas dalam percakapan sehari-hari.

Natural Language Processing (NLP) menjadi sangat penting karena kemampuannya untuk memperbaiki pengalaman pengguna dalam berkomunikasi dengan mesin. Dengan NLP, mesin dapat memahami bahasa manusia dengan lebih akurat dan memproses informasi dengan lebih cepat. Hal ini memungkinkan mesin untuk memperoleh wawasan dari data bahasa manusia, seperti pesan, email, dokumen, atau percakapan di media sosial. NLP juga berguna dalam menjalankan tugas-tugas yang kompleks, seperti penerjemahan bahasa, analisis sentimen, deteksi teks palsu, atau pencarian informasi. Dengan kemampuan ini, NLP dapat membantu meningkatkan efisiensi bisnis, memperbaiki pelayanan pelanggan, meningkatkan produktivitas, dan membantu dalam pengambilan keputusan berbasis data.

Cara Kerja NLP

NLP menggabungkan model linguistik komputasional, machine learning, dan deep learning untuk memproses bahasa manusia. Berikut adalah beberapa cara kerja NLP:

- **Tokenization**: Proses memecah teks menjadi bagian-bagian terkecil yang disebut token. Hal ini dilakukan untuk memudahkan proses analisis dan pemrosesan data teks.
- **NER (Named Entity Recognition)**: Proses mengidentifikasi entitas seperti orang, tempat, atau objek dalam teks. NER menggunakan teknik seperti pos-tagging dan pattern recognition untuk mengenali entitas-entitas tersebut.
- **Sentiment Analysis**: Proses menganalisis sentimen atau perasaan yang terkandung dalam teks. Hal ini berguna untuk mengukur opini atau pandangan orang terhadap suatu topik tertentu.
- **Machine Translation**: Proses menerjemahkan teks dari satu bahasa ke bahasa lainnya. Teknik ini menggunakan algoritma yang mengenal pola-pola bahasa untuk menghasilkan terjemahan yang disesuaikan dengan konteks dan aturan bahasa.
- **Text Summarization**: Proses membuat ringkasan dari teks yang panjang. Teknik ini menggunakan algoritma yang dapat memilih informasi penting dari teks dan menghasilkan versi yang lebih singkat.

Cara kerja NLP sebenarnya sangat kompleks dan melibatkan banyak istilah teknis. Namun, pada dasarnya teknologi ini menggunakan algoritma yang dapat memproses teks secara otomatis sehingga mesin dapat memahami dan menghasilkan output yang berguna bagi manusia.

Langkah-langkah Implementasi NLP

Secara umum, langkah-langkah implementasi Natural Language Processing (NLP) meliputi:

1. **Pengumpulan data**

   Langkah pertama adalah mengumpulkan data teks yang akan diolah. Data dapat berasal dari berbagai sumber seperti media sosial, perusahaan, publikasi ilmiah, atau sumber lainnya.

2. **Pra-pemrosesan data**

Setelah data telah dikumpulkan, langkah selanjutnya adalah melakukan pra-pemrosesan data, yang meliputi penghapusan karakter khusus, stopword, normalisasi teks, dan tokenisasi.

a. Tokenisasi memecah sebuah kalimat menjadi unit kata atau frasa inividual.
b. Stemming dan lemmatisasi menyederhanakan kata ke dalam bentuk akarnya. Misalnya, proses ini mengubah starting menjadi start.
c. Penghapusan stopword memastikan kata yang tidak menambahkan makna siginifikan ke sebuah kalimat, seperti for dan with, dihapus.

3. **Pembuatan korpus**

Korpus adalah kumpulan teks yang memiliki karakteristik dan sifat yang sama. Sebagai contoh, korpus dapat dibuat berdasarkan topik tertentu atau bahasa yang digunakan.

4. **Pengolahan bahasa alami**

Pada tahap ini, data yang telah diproses akan diolah dalam bentuk model atau algoritma untuk memahami makna dan makna dalam teks. Beberapa contoh pengolahan bahasa alami termasuk pengelompokan kategori, analisis sentimen, dan pemodelan topik.

5. **Evaluasi model NLP**

Setelah model NLP dibangun, langkah selanjutnya adalah untuk mengevaluasi performa model. Hal ini dapat dilakukan dengan menggunakan metode evaluasi seperti precision, recall, dan F1 score. Evaluasi bertujuan untuk meningkatkan kualitas model NLP yang dibuat.

6. **Deploy model NLP**

Setelah evaluasi model dilakukan, model NLP dapat di-deploy di berbagai platform, antara lain untuk meningkatkan proses bisnis, memahami keterlibatan pelanggan dan pasar, memonitor media sosial, dan lain-lain.

Aplikasi NLP

Ada banyak aplikasi umum dan praktis NLP di kehidupan kita sehari-hari, antara lain:

- **Asisten virtual -** Aplikasi asisten virtual seperti Siri, Alexa, dan Google Assistant menggunakan NLP untuk memahami dan merespons permintaan pengguna dalam bahasa alami.
- **Penerjemah** - Aplikasi penerjemah seperti Google Translate menggunakan NLP untuk menganalisis konteks dan membuat terjemahan yang akurat.
- **Analisis sentimen** - Aplikasi analisis sentimen seperti Sentiment Analysis dan Hootsuite Insights menggunakan NLP untuk menganalisis dan memahami perasaan dan pendapat yang terkandung dalam teks seperti media sosial, ulasan produk, dan survei.

- **Otomatisasi tugas verbal** - Aplikasi seperti *Dragon Naturally Speaking* dan *Siri Shortcuts* memanfaatkan NLP untuk mengenali perintah verbal dan membuat tugas otomatis.
- **Pencarian informasi** - Aplikasi pencarian seperti Google Search menggunakan NLP untuk memahami pertanyaan dan memberikan jawaban yang relevan.
- **E-mail filtering** - Aplikasi seperti Grammarly dan Boomerang menggunakan NLP untuk memindai, menganalisis, dan membantu pengguna mengedit dan menyaring email.
- **Pengembangan chatbot** - Aplikasi chatbot seperti BotStar dan Tars menggunakan NLP untuk memahami permintaan pengguna dan memberikan tanggapan yang tepat dalam chat.

Dalam keseluruhan, Natural Language Processing memberikan kemampuan untuk sistem komputer untuk memahami bahasa manusia di semua jenis dokumen dan teks. Dengan teknologi ini, aplikasi dapat membantu pengguna memproses informasi dengan lebih efisien dan secara signifikan meningkatkan pengalaman pengguna.

Pembuatan *Chatbot* Sederhana

Pada sesi kali ini, kita akan membuat sebuah Chatbot menggunakan Python. Terdapat beberapa langkah pembuatan Chatbot, yaitu:

1. Menyiapkan korpus

Korpus bahasa Inggris sederhana dapat dilihat pada file “intents.jason” seperti yang terlihat pada Gambar 9.1.

```
{
    "intents": [

        {
            "tag": "google",
            "patterns": [
                "google",
                "search",
                "internet"
            ],
            "responses": [
                "Redirecting to Google..."
            ]
        },
        {
            "tag": "greeting",
            "patterns": [
                "Hi there",
                "How are you",
                "Is anyone there?",
                "Hey",
                "Hola",
                "Hello",
                "Good day",
                "Namaste",
                "yo"
```

Gambar 9.1. Isi file “intents.jason”

Dataset merupakan sebuah objek bernama “intents”. Dataset terdiri dari 30 instan dari “intensts” yang terdiri atas *tag*, *patterns*, *responses*, dan *context*. Sekarang, tugas yang dihadapi

adalah membuat mesin dapat mempelajari pola antara *patterns* dan *tag* sehingga saat pengguna memasukkan sebuah pernyataan, mesin dapat mengidentifikasi *tag* yang sesuai dan memberikan salah satu *responses* sebagai keluaran.

Agar korpus dapat diakses oleh Chatbot, maka tempatkan *file* korpus pada folder yang sama dengan file Python yang kita buat.

2. Mengimpor *library* yang relevan

Simple Chatbhot Using Python

In [28]:
```
#import the relevant libraries
import nltk
nltk.download('punkt')
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import tensorflow as tf
import numpy as np
import tflearn
import random
import json
```

3. Membaca *file* JSON

In [29]:
```
#Reading the JSON file
# We will use json.loads() for this and
# store the content of the file into a variable named 'intents' (type: dict)
data_file = open('intents.json').read()
intents = json.loads(data_file)
```

4. Pra-pemrosesan data

In [30]:
```
#Pre-processing
words = []
classes = []
documents = []
ignore = ['?']
# loop through each sentence in the intent's patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # tokenize each and every word in the sentence
        w = nltk.word_tokenize(pattern)
        # add word to the words list
        words.extend(w)
        # add word(s) to documents
        documents.append((w, intent['tag']))
        # add tags to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

In [37]:
```
# Perform stemming and lower each word as well as remove duplicates
words = [stemmer.stem(w.lower()) for w in words if w not in ignore]
words = sorted(list(set(words)))

# remove duplicate classes
classes = sorted(list(set(classes)))

print (len(documents), "documents")
print (len(classes), "classes", classes)
print (len(words), "unique stemmed words", words)
```

Output:

```
111 documents
30 classes ['Identity', 'activity', 'age', 'appreciate', 'contact', 'covid19', 'cricket', 'datetime',
'exclaim', 'goodbye', 'google', 'greeting', 'greetreply', 'haha', 'inspire', 'insult', 'jokes', 'new
s', 'nicetty', 'no', 'options', 'progname', 'programmer', 'riddle', 'song', 'suggest', 'thanks', 'tim
er', 'weather', 'whatsup']
118 unique stemmed words ["'m", "'s", ',', '10', '19', 'a', 'ag', 'am', 'anyon', 'ar', 'ask', 'aweso
m', 'bad', 'bby', 'be', 'best', 'bye', 'can', 'contact', 'could', 'covid', 'cre', 'cricket', 'cur',
'dat', 'day', 'design', 'develop', 'do', 'doing', 'dumb', 'fin', 'for', 'funny', 'get', 'good', 'good
by', 'googl', 'gre', 'hah', 'hello', 'help', 'hey', 'hi', 'hol', 'hot', 'how', 'i', 'idiot', 'ind',
'inspir', 'internet', 'is', 'it', 'jok', 'kar', 'know', 'lat', 'latest', 'laugh', 'lmao', 'lol', 'los
t', 'mad', 'mak', 'match', 'me', 'mot', 'namast', 'new', 'next', 'nic', 'no', 'nop', 'off', 'ok', 'ol
d', 'program', 'provid', 'quest', 'riddl', 'rofl', 'scor', 'search', 'see', 'set', 'she/h', 'shut',
'song', 'suggest', 'sup', 'support', 'talk', 'tel', 'temp', 'ten', 'thank', 'that', 'the', 'ther', 't
il', 'tim', 'to', 'today', 'top', 'up', 'upto', 'useless', 'was', 'wazzup', 'weath', 'wer', 'what',
'when', 'who', 'yeah', 'yo', 'you']
```

5. Menentukan data *training*

In [32]:
```
# create training data
training = []
output = []
# create an empty array for output
output_empty = [0] * len(classes)

# create training set, bag of words for each sentence
for doc in documents:
    # initialize bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # stemming each word
    pattern_words = [stemmer.stem(word.lower()) for word in pattern_words]
    # create bag of words array
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is '1' for current tag and '0' for rest of other tags
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])

# shuffling features and turning it into np.array
random.shuffle(training)
training = np.array(training, dtype=object)

# creating training lists
train_x = list(training[:,0])
train_y = list(training[:,1])
```

6. Membangun model jaringan syaraf tiruan

In [39]:
```
# resetting underlying graph data
tf.compat.v1.reset_default_graph()

# Building neural network
net = tflearn.input_data(shape=[None, len(train_x[0])])
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, len(train_y[0]), activation='softmax')
net = tflearn.regression(net)

# Defining model and setting up tensorboard
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')

# Start training
model.fit(train_x, train_y, n_epoch=1000, batch_size=8, show_metric=True)
model.save('model.tflearn')
```

Output:

```
Training Step: 13999  | total loss: 0.06624 | time: 0.015s
| Adam | epoch: 1000 | loss: 0.06624 - acc: 0.9875 -- iter: 104/111
Training Step: 14000  | total loss: 0.05998 | time: 0.016s
| Adam | epoch: 1000 | loss: 0.05998 - acc: 0.9888 -- iter: 111/111
--
```

Arsitektur jaringan syaraf tiruan yang dibangun terdiri dari 3 lapisan. Lapisan input terdiri atas 10 neuron/node, lapisan tersembunyi terdiri atas 10 neuron/noden, dan lapisan output memiliki jumlah neuron/node yang sama dengan panjang satu elemen train_y. Fungsi aktivasi yang digunakan merupakan fungsi “softmax”.

7. Mempersiapkan data dan model

In [40]:
```
import pickle
pickle.dump( {'words':words, 'classes':classes, 'train_x':train_x, 'train_y':train_y}, open( "training_data", "wb" ) )
```

In [41]:
```
# restoring all the data structures
data = pickle.load( open( "training_data", "rb" ) )
words = data['words']
classes = data['classes']
train_x = data['train_x']
train_y = data['train_y']
```

In [42]:
```
# load the saved model
model.load('./model.tflearn')
```

8. Membuat fungsi untuk menangani input pengguna

In [43]:
```
def clean_up_sentence(sentence):
    # tokenizing the pattern
    sentence_words = nltk.word_tokenize(sentence)
    # stemming each word
    sentence_words = [stemmer.stem(word.lower()) for word in sentence_words]
    return sentence_words

# returning bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=False):
    # tokenizing the pattern
    sentence_words = clean_up_sentence(sentence)
    # generating bag of words
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)

    return(np.array(bag))
```

In [44]:
```
ERROR_THRESHOLD = 0.30
def classify(sentence):
    # generate probabilities from the model
    results = model.predict([bow(sentence, words)])[0]
    # filter out predictions below a threshold
    results = [[i,r] for i,r in enumerate(results) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append((classes[r[0]], r[1]))
    # return tuple of intent and probability
    return return_list

def response(sentence, userID='123', show_details=False):
    results = classify(sentence)
    # if we have a classification then find the matching intent tag
    if results:
        # loop as long as there are matches to process
        while results:
            for i in intents['intents']:
                # find a tag matching the first result
                if i['tag'] == results[0][0]:
                    # a random response from the intent
                    return print(random.choice(i['responses']))

            results.pop(0)
```

Adapun penjelasan singkat dari fungsi-fungsi di atas adalah sebagai berikut:

a. Fungsi **clean_up_sentence(sentence)** : Fungsi ini menerima *sentence* (string) sebagai input dan kemudian membuat token menggunakan *nltk.word_tokenize()*. Setiap token kemudian diubah menjadi bentuk akarnya menggunakan *stemmer*. Outputnya pada dasarnya adalah daftar kata-kata dalam bentuk akarnya.

b. Fungsi **bow(sentence, words, show_details=False)** : Fungsi ini memanggil fungsi **clean_up_sentence(sentence)**, mengonversi teks menjadi *array* menggunakan model bag-of-words menggunakan kosakata masukan, lalu mengembalikan nilai *array* yang sama..

c. Fungsi **classify(sentence)** : Fungsi ini mengambil *sentence* sebagai input dan mengembalikan *list* yang berisi tag yang sesuai dengan probabilitas tertinggi.

d. Fungsi **response(sentence, userID='123', show_details=False)** : Fungsi ini menerima *tag* yang dikembalikan oleh fungsi **classify(sentence)** dan menggunakannya untuk memilih respons yang sesuai dengan *tag* yang sama di “intents.json” secara acak.

9. Interaksi dengan Chatbot

```
In [*]: print("Press 0 if you don't want to chat with our chatbot.")
        while True:
            message = input("")
            if message == "0":
                break
            result = response(message)
            print(result)
```

Output:

```
Press 0 if you don't want to chat with our chatbot.

Press 0 if you don't want to chat with our chatbot.
Who are you?
I am Sim, a simple chatbot
None

Press 0 if you don't want to chat with our chatbot.
Who are you?
I am Sim, a simple chatbot
None
Goodbye
Have a nice day
None
0

Press 0 if you don't want to chat with our chatbot.
Who are you?
I am Sim, a simple chatbot
None
Goodbye
Have a nice day
None
0
```

Latihan: Mengubah korpus ke dalam Bahasa Indonesia

1. Ubah korpus (kumpulan teks) dalam file “intents.json” menjadi bahasa Indonesia!
2. Lakukan proses pelatihan ulang!
3. Bagaimana respon yang diberikan oleh chatbot dengan menggunakan korpus berbahasa Indonesia? Tunjukkan dan Jelaskan!

Referensi:

https://www.projectpro.io/article/python-chatbot-project-learn-to-build-a-chatbot-from-scratch/429#mcetoc_1fofi4362b
https://github.com/Karan-Malik/Chatbot/blob/master/chatbot_codes/intents.json
https://www.sas.com/id_id/insights/analytics/what-is-natural-language-processing-nlp.html
https://aws.amazon.com/id/what-is/nlp/#:~:text=Pemrosesan%20bahasa%20alami%20(NLP)%20adalah,memanipulasi%2C%20dan%20memahami%20bahasa%20manusia.